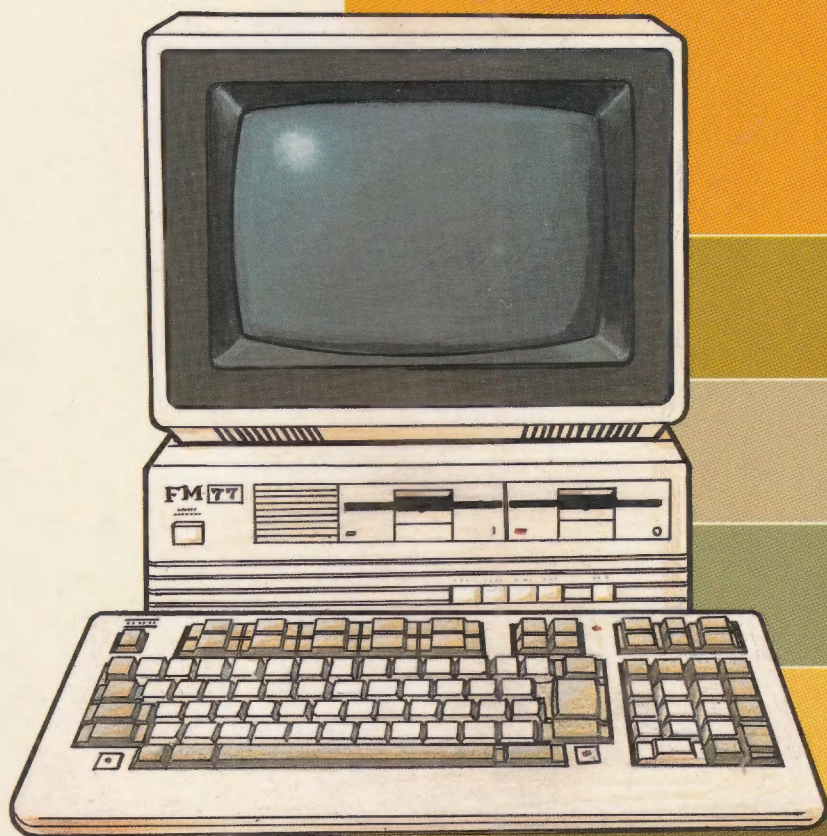
知っておきたい83のポイント集

FM-77の本

上柿　力●著

BASIC, Logoプログラミング

ラジオ技術社

知っておきたい83のポイント集

FM-77の本

BASIC, Logoプログラミング

上柿　力　著

ラジオ技術選書148

FM-77の本 — BASIC, Logoプログラミング

◆目　次◆

上柿　力　著

1 これだけは知っておきたいFM-77のABC

2 本格的なプログラム作りのために

3 すっかり分るファイルの操作

4 グラフィックスをマスターしましょう

序

FM-8から始まった富士通の8ビット・パーソナルコンピュータは，FM-77で事実上，最終モデルになったといえるでしょう．多機能性・高速性を備え，豊富なオプションの活用などにより，実用性が高く楽しく使えるものになったからです．また，BASICやLogoに加えてOS-9なども導入できるため，幅広い利用価値を持っています．

このようにすぐれた能力を持つFM-77ですが，いざ自分のものになり，これを使って何かをさせようというとき，そのすべてを市販のソフトウェアに頼るわけにはゆきませんし，プログラミングの基本になる知識なしでは使いこなせません．また，身の回りにあるいろいろなテーマの多くは，ちょっとしたプログラムを自分で作ることで処理できるものです．

本書はこうした観点から，コンピュータについて基礎的な理解を必要とする部分についての解説とともに，実際的な「FM-77をどのようなことにどう使うのか」に関するテーマを，多くのプログラムリストを例にしてまとめたものです．したがってBASICやLogoの文法そのものの解説などは割愛していますから，製品に添付されたマニュアル類を参照しながら本書に目を通してくだされば，より深く正しく理解できることと思います．

本書を読まれるにあたって，以下の点について注意してください．

- BASICはF-BASICのV3.0に準拠しています
- LogoはFM-LogoのV2.0に準拠しています
- BASICのプログラムリストの末尾には，REM（リマーク）あるいはその代用であるアポストロフィ（′）に続き，解説の文を入れてあります．プログラムを走らせる上では不要ですから，実使用にあたっては省いてもさしつかえありません
- 内容の大半はFM-7およびFM-new7にも対応します．なお，BASICについては漢字ROMやフロッピーディスク・ドライブを装備することで完全に対応します

1984年9月

（3台のFM……11, 7, 77に囲まれて）

◆ 参考文献

マイクロコンピュータの内部構造と機械語　福永邦雄著（CQ出版）

BASIC入門／活用コース　T.ドワイヤー／M.クリッチフィールド著（現代数学社）

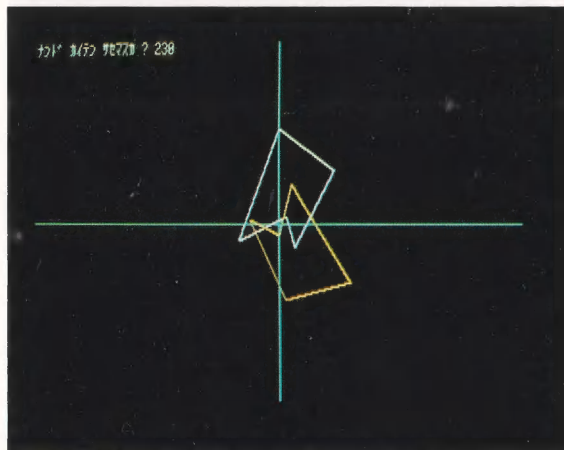
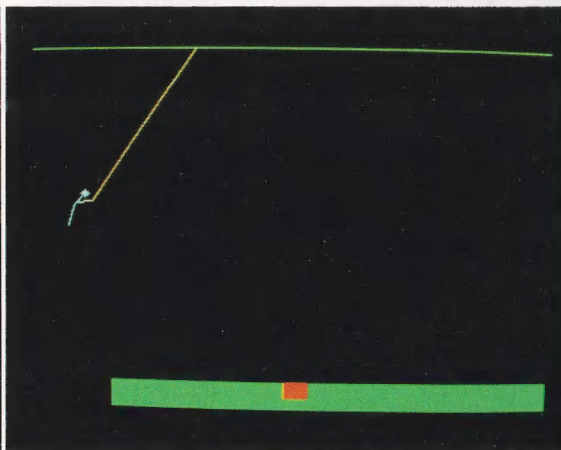
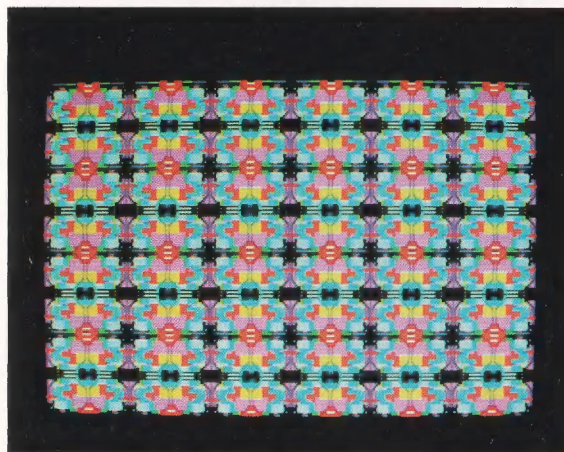
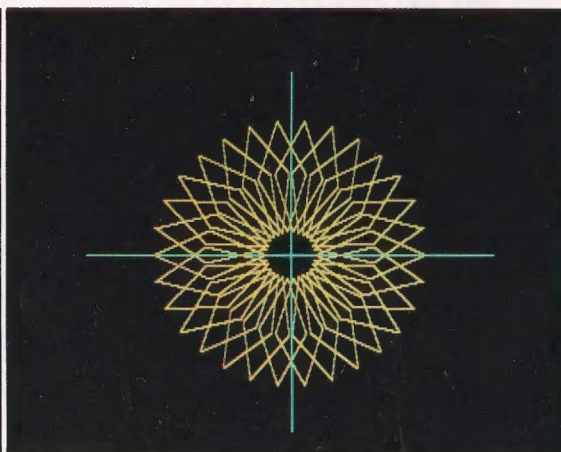
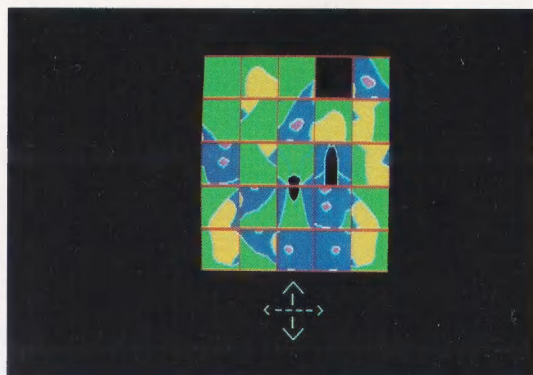
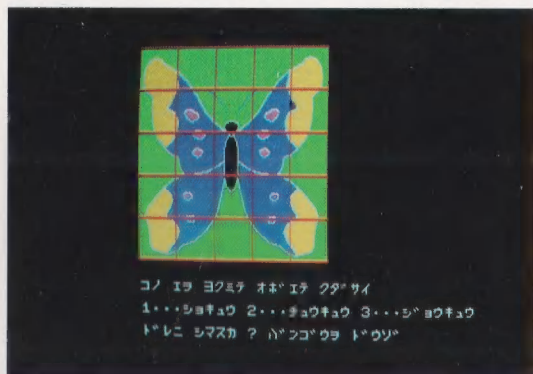
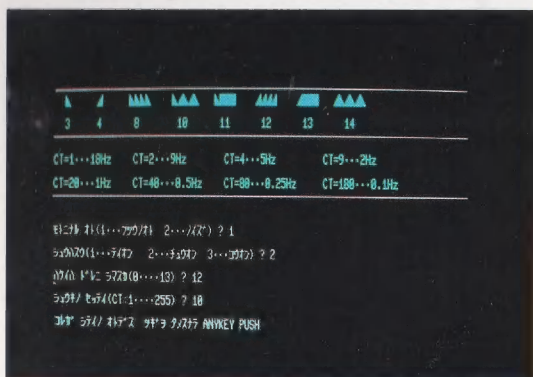
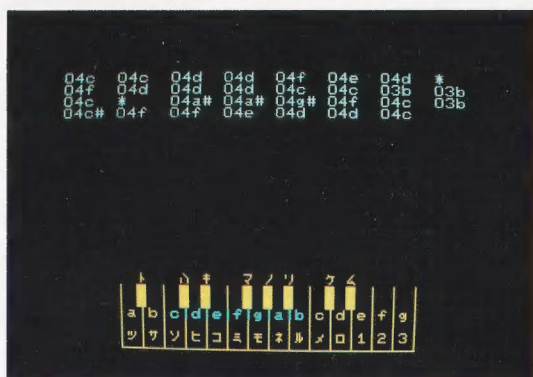
マイコン／ビジネスソフト1年生　西尾文明著（ラジオ技術社）

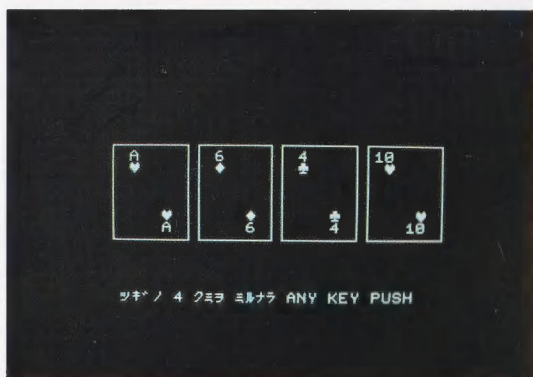
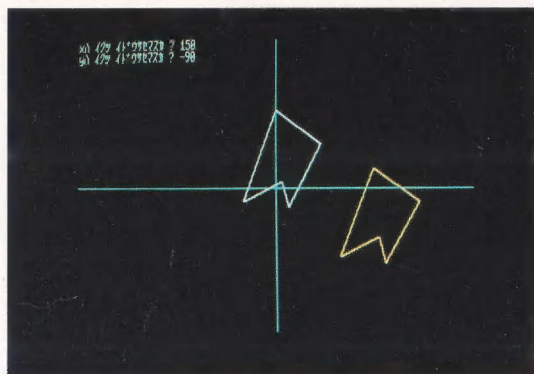
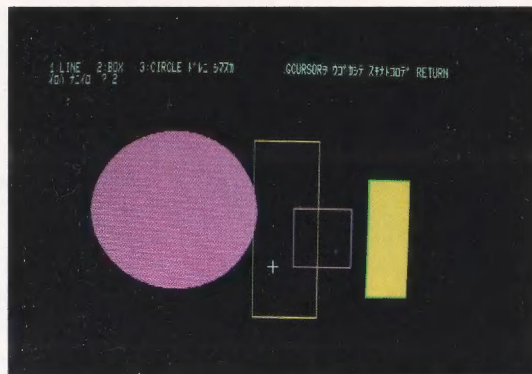
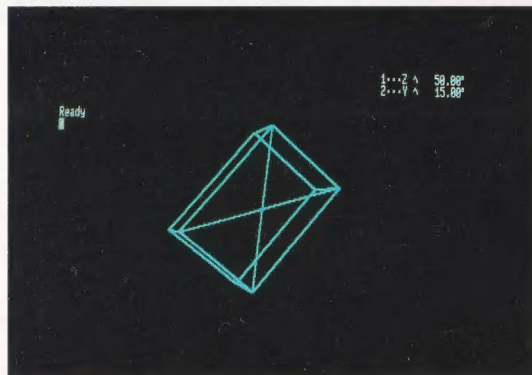
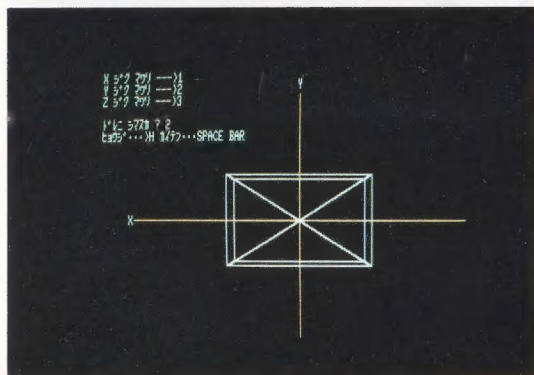
パソコン実用ガイド '83　（ラジオ技術社）

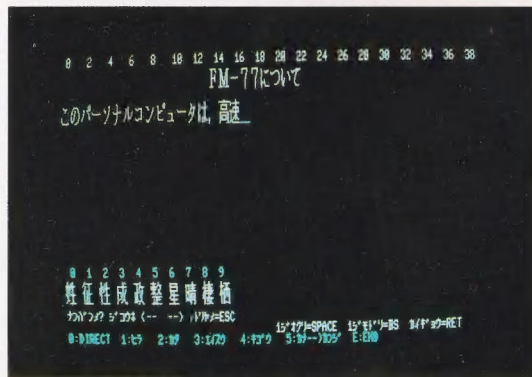
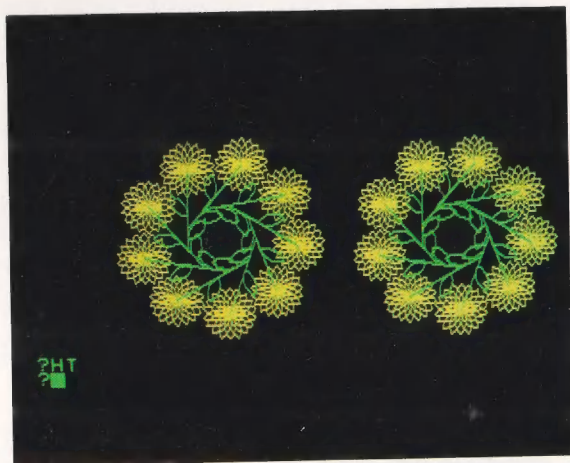
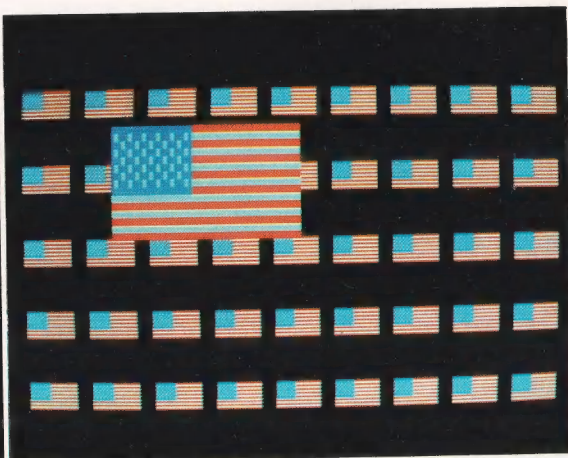
図形科学ハンドブック　日本図学会編（森北出版）

現代図学　小高司郎著（森北出版）

マインドストーム　S.パパート著／奥村貴世子訳（未来社）

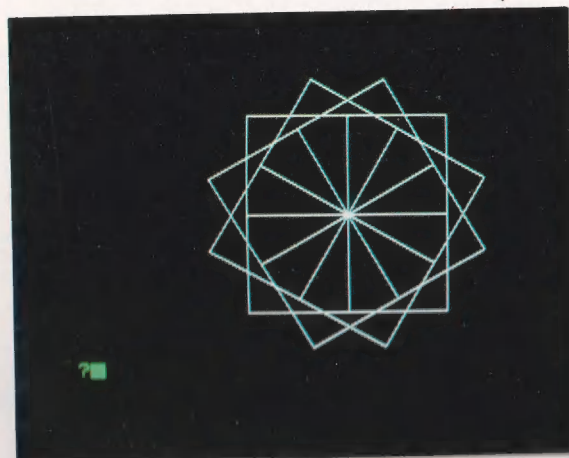
BASICでつくるグラフィックス

図形を回転させてみる(105ページ)

GET@を使った空中ブランコ・アニメーション(130ページ)

きれいなカレイドスコープ(万華鏡)

ひし形を回転移動させてみる(107ページ)

24枚の絵合せゲーム(133ページ)

波形加工をして効果音を出してみる(159ページ)

FM-77が鍵盤楽器になる(169ページ)

カラーパレットをテストしてみる(120ページ)

配列をトランプで考えると…(50ページ)

図形の移動(95ページ)

グラフィックカーソルで絵を描く(118ページ)

立体の回転．左は最初の画面．右は回転後の表示(113ページ)

漢字のフォントを拡大して見る(154ページ)

漢字カナまじり文で文書を作る(149ページ)

複雑な花模様もかんたんに作れる(183ページ)

シェイプを使って旗を描く(185ページ)

Logoの世界に入ってみよう

四角形を回転させるのもかんたんにできる(176ページ)

Logoでしりとりあそびをしよう(192ページ)

```
TO カウント
PR ( SE [デ゛タコトハ゛ノカズ゛ ハ] COUNT :デ゛タコトハ゛ I
[デ゛シタ] )
END

TO ツナカ゛リマセン
PR [ ]
PR [シリトリニ ナッテイマセン]
カウント
END

TO ンデ゛オワリ
PR [ ]
PR [ンデ゛オワッテイマス ゲ゛ームセット]
カウント
END

TO モウデ゛マシタ
PR [ ]
PR [ソノコトハ゛ハ モウデ゛マシタ]
カウント
END

TO ツギ゛ノコトハ゛
```

1

これだけは知っておきたい

FM-77のABC

ディスケットのフォーマット

＊フォーマットとは

フロッピーディスケットは，A社にもB社のコンピュータにも使えます．しかし場合によってはA社とB社ではディスケットの使いかたが異なるため，新品のディスケットは畑でいえば“荒地”の状態になっています．これを，使用するコンピュータに合った形に整地するのがフォーマット（format：一定の方式にする）です．新品のディスケットを1枚用意して，まずマニュアルにしたがってフォーマットしてください．これが終ったら，とりあえずシステムディスケットをバックアップ（backup：予備品を作る）しましょう．こうしておけばマスターになるディスケットはとりあえず不要になるので，これをたいせつに保管しておきます．

＊＊オリジナルのディスケットは大切に保管

つぎに WIDTH 80にして，リストを画面に出したら，このフォーマット用のプログラムを，下のように書き変えましょう．ディスケットのフォーマットはひんぱんに行ないますが，オリジナルプログラムは分りにくいので，日本語に直すのが大部分です．行番号545，546，547は追加する文で，これを書いておけばマニュアルをいちいち参照しなくてもフォーマットが正しく行なえます．リストを書き変えたら

SAVE "0：SYSDSK ↵

としましょう．

Are you sure (Y or N)?

で，Yを入力して終りです．

※ディスケットに書かれたデータやプログラムのすべてを消したい時は，ディスケットをイニシアライズする
DSKINI 0
といった命令ですべてが消去可能

```
10 '
20 '     DISK FORMAT
30 '
40 CLEAR 10000,&H4FFF
50 LOADM"DFMCD"
60 WIDTH 80,20:CLS:COLOR 4:PRINT
70 PRINT"*** FLOPPY DISK FORMATTING ***"
80 COLOR 7:PRINT
90 INPUT"ﾌｫｰﾏｯﾄﾃﾞｽｶ ";R$
100 IF R$<>"Y" GOTO 220
110 INPUT"ﾌｫｰﾏｯﾄﾖｳﾉ ﾃﾞｨｽｹｯﾄﾗ ﾅﾝﾊﾞﾝﾉ ﾄﾞﾗｲﾌﾞﾆ ｲﾚﾏｽｶ(0-3)";DRV2
120 IF DRV2<0 GOTO 110
130 IF DRV2>3 GOTO 110
```

```
140 PRINT "ﾌｫｰﾏｯﾄｼﾀｲ DISK ｦ"
150 PRINT"ﾄﾞﾗｲﾌﾞ";DRV2;"ﾆ ｲﾚﾃｸﾀﾞｻｲ"
160 INPUT"ﾖｳｲ OK ";R$:IF R$<>"Y" GOTO 110
165 POKE &H5400,0
170 POKE &H5401,DRV2
180 EXEC &H5000
190 E=PEEK(&H5400)
200 IF E<>0 THEN PRINT"DISK I/O ERROR !!":GOTO 540
210 PRINT "ﾌｫｰﾏｯﾄ ｶﾝﾘｮｳ !!"
220 PRINT
230 '
240 'DISK BASIC SYSTEM COPY
250 '
260 DIM S$(63)
270 FIELD #0,128 AS A$,128 AS B$
280 COLOR 4
290 PRINT"*** DISK BASIC SYSTEM ｦ COPY ｼﾏｽ ***"
300 COLOR 7:PRINT
310 INPUT "ﾅﾝﾊﾞﾝﾉ ﾄﾞﾗｲﾌﾞﾆ ｵｸﾘﾏｽｶ(0-3)";DRV2
320 IF DRV2<0 OR DRV2>3 THEN 310
330 PRINT"ｼｽﾃﾑ  DISK ｦ DRIVE 0 ﾆｲﾚﾃｸﾀﾞｻｲ"
340 IF DRV2=0 THEN 360
350 PRINT"ｺﾋﾟｰ ｼﾀｲ ﾃﾞｨｽｹｯﾄｦ DRIVE";DRV2;"ﾆｲﾚﾃｸﾀﾞｻｲ"
360 DRV1=0
370 INPUT"ﾖｳｲ OK ";R$:IF R$<>"Y" GOTO 310
380 FOR SCT=1 TO 32
390 DUMMY$=DSKI$(DRV1,0,SCT)
400 SAD=2*(SCT-1)
410 S$(SAD)=A$
420 S$(SAD+1)=B$
430 NEXT SCT
440 IF DRV2<>0 GOTO 470
450 PRINT"ｺﾋﾟｰ ｼﾀｲ ﾃﾞｨｽｹｯﾄｦ DRIVE";DRV2;"ﾆｲﾚﾃｸﾀﾞｻｲ"
460 INPUT"ﾖｳｲ OK  ";R$:IF R$<>"Y" GOTO 450
470 FOR SCT=1 TO 32
480 SAD=2*(SCT-1)
490 LSET A$=S$(SAD)
500 LSET B$=S$(SAD+1)
510 DSKO$ DRV2,0,SCT
520 NEXT SCT
530 PRINT "SYSTEM CODE ｺﾋﾟｰ ｶﾝﾘｮｳ !!"
540 CLEAR 300,&H6FFF
545 PRINT"ｲﾏ COPY ｼﾀﾓﾉｦ ｱﾀﾗｼｸ ﾂｶｳﾄｷﾊ"
546 PRINT"DSKINI ﾆ ﾂﾂﾞｷ,ﾃﾞｨｽｹｯﾄｶﾞ ﾊｲｯﾃｲﾙ ﾄﾞﾗｲﾌﾞﾉ ﾊﾞﾝｺﾞｳｦ ﾀｲﾌﾟ"
547 PRINT"ﾚｲ <<DSKINI0>> ﾔ <<DSKINI1>> ﾅﾄﾞﾃﾞｽ"
550 END
```

※このプログラムはFM-77付属のユーティリティをもとにしています

Logoのシステムディスクもバックアップを

＊Logoのバックアップ

Logoのシステムディスク（システムディスケット）も，おなじようにバックアップしましょう．まずF-BASICのシステムディスクをドライブの０番に入れ

```
RUN "VOLCOPY ⏎
```

とします．しばらくすると画面にメッセージが出ますから，以下，画面の指示に従います．なお，このVOLCOPY（volume copy：ディスケットの中味をそっくり他のディスケットへ写すこと）などのユーティリティプログラムも，日本語の説明に書きかえたほうが使いやすくなります．前のページを参照して試してください．ちなみに

SOURCE：ソース／元のもの

DESTINATION：デスティネーション／目的のもの，行き先

ということです．ボリュームコピーは５トラック分をひとまとめにして，何回かに分けて目的のディスケットへと写していきます．

※画面にメッセージが出たら，F-BASICのシステムディスケットを取出し，代りにLogoのシステムディスケットをドライブ０番に入れ，あとは画面の指示にしたがう

＊＊最小区画はクラスタ

ディスケットに収めることができるものはプログラムと，プログラムで作成したデータです．これらはすべてファイルと呼ばれますが，プログラムファイルやデータファイルはディレクトリ（directory：管理，管理名簿を見ること）をとって，一覧表の形で画面に出力することができます．このために使うのがファイルス命令で

```
FILES "0: ⏎
```

というようにタイプすることで可能です．画面出力されたファイルの読み方についてはマニュアルを参照してください．

なおプログラムもデータも，ディスケットの最少区分単位であるクラスタ（cluster：ひとたば，ひとかたまり）を基準にして収められ，DSKINIがほどこされたディスケットでは152クラスタの区画が用意されています．たとえば

```
100 A=10：PRINT A
```

というプログラムはとても短かく，本質的にはディスケットのほんの一部に収められるので，１クラスタといった区画の中ではゴミつぶ程度の存在ですが，プログラムファイルとしてSAVEすれば，やはり１クラスタを使ってしまいます．ためしに上のプログラムを作り

※ディスケットの入／出力最小単位はセクタ．ただしプログラムファイルなどは８セクタを１クラスタとし，これを管理の最小単位としている

```
FOR I=1 TO 152：SAVE "0："+STR$(I)：NEXT
```

として，リターンキーを押してください．FM-77は数分かけてこのプログラムを152回，SAVEしてゆきます．

こののちにディレクトリをとると，プログラム名が"1","2",……とつけられたものが152個記録されていることを教えてくれます．短かいプログラムであっても，1枚のディスケットには最大で152までしか収められない，というわけです．

***よいファイル名，悪いファイル名

あなたが作るプログラムやデータファイルには，名前をつけて管理しなければいけません．1枚のディスケットの中にデータ，BASIC，Logoのプログラムを収めることができますが，BASICとLogoは別のディスケットを使いましょう．やむなく混在させる時は

アテナカキ．BC　(BASICによる宛名書きプログラム)

アカイハナ．LG　(Logoによる赤い花を描くプログラム)

などと，ファイル名のあとに識別記号を書いておきます．F-BASICではファイル名は8文字までが使えます．なお，プログラムファイルか，データファイルかはディレクトリをとって見ることができるので識別記号を書く必要はありません．

プログラムを作ってSAVEするときに，たとえば

"TEST1"，"TEST2" ……

"PROG1"，"PROG2" ……

といったような名前をつけることは，一見するととても合理的なのですが，それらがたくさんになってくると，ディレクトリをとっても内容が分らなくてこまってしまうことになります．プログラムの内容が分かるファイル名にしましょう．また，プリンタがあるのなら必らずリストを打出しておき，保存しておきます．

また，作成したプログラムには作成者の名前，作成日時などをレム文(REM：リマーク)にしてプログラムの最初に書いておきます．こうしておけばいちいちFM-77に電気を入れなくても，プログラムリストを見ることである程度の管理ができるようになります．なお，ディスケットにも通し番号をつけると共に，簡単なメモ帳を入れておき，それぞれのディスケットには何が入っているのかを書いておきましょう．ディスケットが3枚や5枚程度なら管理もむずかしくありませんが，20枚，30枚になると覚え書きなしでは管理できません．

FM-77の10進数，ビット表現

＊FM-77と整数の扱い

現在のコンピュータはディジタルコンピュータといって，電気信号の“ある”“なし”を利用していることはごぞんじですね．FM-77に限らず，8ビットのパーソナルコンピュータは2^{16}（2の16乗(じょう)）＝16ビットで整数を扱うことができます．2の16乗は65536になりますが，もしマイナスの数を扱わなくてもよいのなら，この65536通りを0，1，2……，65535という整数に対応させてもよいといえます．しかし実際にはマイナスの数を扱わないことには，計算機として使いにくいために，65536通りをつぎのように対応させます．

－32768，－32767，……，－1，0，1，……，32766，32767

＊＊“補数”表現とは

FM-77が扱う整数の最小値は－32768，最大値が32767になっているわけですが，16ビットの信号の“ある”，“なし”の「並び」はどうなっているのかといいますと，“ある＝1”“なし＝0”の表現で

－32768	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
－32767	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
－5	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
－2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
－1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
5	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
32766	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
32767	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

などとなります．これを見て気がつくことは

－32768	1̇ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
32767	0̇ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

など，マイナスは最上位（左はし）が常に1で，あとはある法則にしたがって，正の数と負の数は反対に（0は1に，1は0になる）なっていることです．これを「補数表現」というのですが，この方法によりコンピュータ自身が計算しやすい仕組になっています．

※ビット（bit）：バイナリディジット（binary digit）と同じで，1ビットは《イエス・ノー》とか《1・0》といったように2つの状態のどちらかであることを意味する．情報の最小単位ともいう．

＊＊＊10進数のビット表示を知るには

コンピュータが扱う数をなぜ16ビットの並びで考えなければいけないのか，あるいはどうしてこんなめんどうなことを説明するのかというと，これから出てくる「論理演算」と関係があるからです．

2　AND　8

などといった論理演算の結果はFM-77によってすぐに求めることができますが，なぜ結果が「0」になるかなどを確める時，いまの16ビットによる整数の表現が使えるのです．

下に示すプログラムは10進数を16桁のビットで表示するものです．ある数を2進数（0と1だけを使って数を表わす）で表現するには，下のリストの行番号240のように，ある数（X）を2で割った余り（B）がその桁のビットになります．そしてある数（X）を2で割る（¥を使った整数除算）というくりかえしになります．たとえばある数が4なら，まず2で割ったときの余りは0，4÷2＝2（整数除算），2を2で割った余りは0，2÷2＝1（整数除算），1を2で割ったときの余りは1，1÷2（整数除算）は0……なので配列変数のB(1)＝0，B(2)＝0，B(3)＝1，B(4)＝0……これを行番号290のように，逆に表示すると16ビットの表示になります．

※このプログラムにはマイナスの整数を補数表現するための特別な工夫がされている

```
100 '================== 10ｼﾝ････>2ｼﾝ ﾍﾝｶﾝ ==================
110 '     ｺﾉﾍﾝｶﾝﾊ FM-77ﾆｵｹﾙ ｼﾞｯｻｲﾃｷﾅ ｶｽﾞﾉ ﾄﾘｱﾂｶｲ ｦ
120 '                         ｾﾂﾒｲｼﾃｲﾏｽ
130 '=====================================================
140 '
150 DEFINT A-Z '････････････････････････････ ｾｲｽｳｶﾞﾀ ｦ ｾﾝｹﾞﾝ
160 WIDTH 80,25
170 DIM B(16) '････････････････････････････ 16ｹﾀﾌﾞﾝｦ ﾖｳｲ
190 F=0
200     INPUT"10ｼﾝｽｳｦ ﾄﾞｳｿﾞ";X:PRINT USING "###### = ";X;
210     IF SGN(X)=-1 THEN F=1:X=X+1:'････ ﾏｲﾅｽ ﾉ ｶｽﾞﾉ ｼｮﾘ
220 '
230  FOR KETA=1 TO 16 '･････････････････････ 16ｹﾀﾌﾞﾝ ｦ ｼﾞｯｺｳ
240    B(KETA)=X MOD 2 '･･････････････････ ｱﾏﾘｦ ﾓﾄﾒﾙ
250     IF F THEN B(KETA)=B(KETA)+1:'････ ﾏｲﾅｽﾉ ｶｽﾞﾉ ｼｮﾘ
260    X=X¥2 '････････････････････････････ ｼｮｳ ｦ ﾓﾄﾒﾙ
270  NEXT KETA
280 '
290 FOR KETA=16 TO 1 STEP -1 '･･････････････ ｼﾞｮｳｲﾉ ｹﾀｶﾗ ﾋｮｳｼﾞ
300    PRINT B(KETA);
310 NEXT KETA
320 PRINT
330 '
340 GOTO 190
```

10進数を16ビットで表示する▶

16進数はコンピュータにとって便利なもの

*電気の世界と16進数

人間にとって10進数はとても区切りがよく，理解しやすいものですし，小さいときから10進数で教育を受けていることもあって，10進数による表現が複雑でもおどろいたりしません．ところがコンピュータときたら，なにしろ電気の信号の ある・なし，しか理解できないものですから，コンピュータから見ると2進数がもっとも分りやすいものなのです．つまり1と0だけで数を表わしたものは，たとえどんなに桁数が増えてもコンピュータはおどろかないのです．

さきほど10進数を2進で表示をした例を見ても分るように，10進数なら5ケタの数字でよいものも，2進数では16ケタを使って表わさないといけない，といったあんばいです．というわけで，コンピュータに関係する技術者・学者は2進数だけではなく16進数というものをよく使います．

**0～Fまでを使って表現する

この16進数はコンピュータを扱うのにとても便利，というわけですが，これも慣れない人間にとってはなかなか理解しにくいものです．しかしFM-77を使いこなす上で，2進数と共に16進数をよく知っておくことはとても役に立つことです．FM-77では10進数も2進数も16進数もすべて使えますが，FM-77にとっていちばん理解しやすいのは，おそらく16進数でしょう．

16進数はその名のとおり，0から始まり15になって，つぎの16になるときにひとケタ上がるものです．ただし，いま“15”といったのは10進数で表現したときの話で，じっさいには

0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F

というかぞえかたをします．ですから16進数の“12”は“じゅうに”ではなく“いちに”になります．また10進数との区別のために&Hをつけて“&H12”と表わすと，これは

下1ケタ目は2なので2個

下2ケタ目は1なので（10進数で1が10個集まると下2ケタ目が1になるように）16個

16個＋2個＝18個（10進表示）

ということです．ためしに

※FM-77で扱える16進数は&H0から&HFFFFまで．それを越えるとエラーになる

PRINT　&H12 ↵
とすると，FM-77は10進表示で「18」を返してきます．

＊＊＊4ビット分が16進数のひとけたに当る

いっぽうコンピュータは1ビットを最小単位としていますが，1ビットの単位で動作することはほとんどありません．もちろんこまかい指定のときは1ビットずつ，ということはありますが，4，8ビットというのがよく使われます．4ビットは（$2^4=16$）で，ちょうど16進数の1ケタ分に対応します．4ビットはしたがって，0～Fまでの字を使った16進数とピタリと合うのです．

0001＝1　1000＝8　1010＝A　1111＝F
というようにです．

いっぽう，8ビット（$2^8=256$，0から始めると255）では16進数の表示では最大がFF……つまり2ケタになります．こうして16進数と4，8，12，16ビットがピッタリと対応するために，なにかと使われるのです．見れば見るほど16進数は頭がクラクラしますが，要は慣れの問題といえるでしょう．

&HC00F
のつぎに大きい数（つまり&HC00F＋&H1）はいくつでしょうか？下1ケタ目のFがひとつくりあがって，下2ケタ目の0が1になり

&HC010
になります．では&HC000よりひとつ少ない数は？　上3ケタがすべてくりさがってくるので，すべてのケタが変わり

&HBFFF
になります．

＊＊＊＊FM-77を有効に使おう

FM-77にはある数を入力すると16進数で結果を返す関数があり

PRINT　HEX$(3128) ↵
PRINT　HEX$(&HF012＋&HFF) ↵
といったように使えます．また，16進数を10進数に直すときは

PRINT　&HF111 ↵
とすれば，ちゃんと10進数に変換してくれます．なお，注意しなければいけないのは負の数と16進数表示の対応です．

PRINT　HEX$(−65536) ↵
はエラーではありません．結果は0になります．なお実際には負の16進数を使うことはまずありません．

人間の判断，コンピュータの判断

＊3 AND 5とは？

FM-77における論理演算については「F-BASIC文法書」に載っています．真理値表とよばれる，AND, OR, XORその他の演算の原理が分りますから前のページで述べた，10進数を16ビットの2進数で表現したものとの組合せて求めることができます．なお，断わるまでもありませんが「3　AND　5」とは「3であり5でもある数」ということではありません．ANDは1を真，0を偽として

〔1・0〕＝0　〔1・1〕＝1　〔0・1〕＝0　〔0・0〕＝0

になるものです．では「3　AND　5」を求めましょう

```
3      0000000000000011  ↓ (それぞれのビット単
5      0000000000000101    位でANDを演算)
  AND  0000000000000001
```

論理演算はビット同士の加算や減算とはちがうので，それぞれのケタはくり上がりなどはしません．「3 AND 5」は

0000000000000001

になりました．10進数でいえば1というわけです．

＊＊真か偽か……それが問題だ

コンピュータは内部での計算や判断など，すべての結果を「真」か「偽」かのどちらかで判断します．IF文による処理は，すべてFM-77が「真」と判断するか「偽」と判断するかによっているわけです．使

```
100 '================= ■ OR B ====================
110 '
120 DEFINT A-Z
130 PRINT
140 PRINT "----------------------"
150    INPUT "A  ";A
160    INPUT "B  ";B
170 PRINT
180        PRINT "A=";(A>=5);":";"B=";(B<=3);"  ";
190     IF A>=5 OR B<=3 THEN PRINT "= TRUE":GOTO 210
200     PRINT "= FALSE"
210 PRINT "----------------------"
220 '
230 GOTO 130
```

```
----------------------
A  ? 1
B  ? 5

A= 0 :B= 0   = FALSE
----------------------

----------------------
A  ? 8
B  ? 4

A=-1 :B= 0   = TRUE
----------------------
```

▲A,Bの2つの数を入力し，ORの条件に合うかどうかを判断させる

う人間が頭で描く「正しい・まちがっている」という判断とはまったく関係なく，コンピュータにとっての「真・偽」が前提になっていることをお忘れなく．

コンピュータが処理をした結果は，BASICではPRINT文で求めることができます．

結果が0のとき……偽

結果が0以外のとき……真

これがFM-77の判断です．たとえば「5は3よりも大きい」というのはまぎれもなく正しいことですが，FM-77はどう判断するでしょうか？

PRINT 5 > 3 ↵

−1

FM-77の結果は−1と出ました．0ではありませんから「真」です．では（2 XOR 4） AND （4 OR 5）は？

これも

PRINT (2 XOR 4) AND (4 OR 5) ↵

で求めることができます(結果は4で真)．

PRINT 2 XOR 4 AND 4 OR 5 ↵

はどうなるでしょうか？（演算の優先順位によって結果は別です）．

※ ↵ はリターンキーを押すことを表わしている

***判断の落し穴にひっかからないようにご注意を

左のページのリストはAとBという，2つの変数がIF文の条件を満たしていれば「TRUE」を出力するものです．下のリストは日本語の感覚では「Aが5以上か，10以下なら」真である．つまりAが5，6…10のときに真になるように思えますが，本当でしょうか？

```
100 '================= A OR A =====================
110 '
120 DEFINT A
130 PRINT
140 PRINT "-----------------------"
150    INPUT "A  ";A
160         PRINT "A=";(A>=5);":";"A=";(A<=10);"  ";
170      IF A>=5 OR A<=10 THEN PRINT "= TRUE":GOTO 190
180      PRINT "= FALSE"
190 PRINT "-----------------------"
200 '
210 GOTO 130
```

この条件を満たす数Aは▶いくつでしょうか？

すべての式は記号で与えられる

＊演算子(オペレータ)とは？

コンピュータは判断をし，計算をします．電卓とコンピュータの決定的な違いは，判断をさせることができるかどうかにある，といってもよいでしょう．ところでこの判断や計算などを具体的に命令するものは何でしょうか？ それらはすべて記号です．この記号のことをコンピュータの世界では演算子（オペレータ：operator）といいます．オペレータ（操作をする者，操作命令記号）を“演算”と訳したのは誰なのでしょうか．演算というと私たちはとかく“計算”と同じだと考えてしまいます．

コンピュータに2つ以上の条件を与えるときは，かならずオペレータを使わないといけません．加算を命令する「＋」もオペレータですし，大小の比較をする「＜」もオペレータです．もちろん論理演算をさせるときもオペレータを使います．

＊＊式と演算子

これらのオペレータを使った表現を一般に式といいます．したがってFM-77における式はじつに幅の広いものになります．

```
3+5    A=8    B=B*B    A$="77"
A<8    X AND Y    M=INT(A/B)
```

などはみな式です．コンピュータにおける式はすべて文字か数字で返ってきます．もしAという変数にじっさいには100という数値が代入されているとき（A＜8）という式が与えられたとしましょう．コンピュータはコンピュータ自身の内部で“これはウソだ”とか“これはまちがっている”などと思っているわけではありません．

式の結果について私たちが知りたいと思ったとき，それを直接的に知ることができる命令は「PRINT」です．算術的な式ではコンピュータは算術の法則にしたがって，その結果を返します．

```
PRINT 3+5 ↵
```

なら「8」という数字を出力します．PRINT文をよく理解していないと「PRINTは文字を画面に出したり，四則演算などの算術的な計算の答を画面に出す命令」と思いがちですが，それだけではないのです．PRINT命令はすべての式の結果，メモリに代入されている変数の現在の状態（数値や文字）などを出力する重要な命令であることをおぼえ

ておきましょう．プリント命令があるのとないのとでは，まるで違うことになる例を紹介します．

PRINT　600／2 ↲

これは画面に「300」が出ます．では

600／2 ↲

とすると，どうなるでしょうか？FM-77は何もしない？いいえ，FM-77はあなたの命令にしたがってちゃんと受付けます．

＊＊＊FM-77にはどんな式があるのか

では式を整理しましょう

●**算術式**（算術演算子を使ったもの）

これは学校で習った算数とおなじように数学的な計算をするものですから，詳しくは説明しません．ただし注意点がいくつかあります．まず割る数を0にしてはいけません．また，計算の結果は誤差が出ることがあります．たとえば2^{13}を計算させてください．答は小数を含んだものになります．算術式の立てかたに注意しましょう．

●**論理式**（論理演算子を使ったもの）

これは別の項目でお話しました(18ページ参照)．

●**関係式**（関係演算子を使ったもの）

＜，＞，＝　と，その組合せによるもので，比較式とか比較演算式などと呼ばれることがあります．この式で私たちがつまづくのは，日本語ではよく「以上」「以下」という言葉を使うのに対して，「○○より大きい」「○○より小さい」に弱いことです．あるいは「より小さい＝未満」という言葉があるのに対して「より大きい」に対応する2文字の熟語がないこと，ともいえます．＜や＝を使って判断文を書くときには十分に注意してください．ちなみにFM-Logoには「＜＝」(日本語でいえば以上，以下にあたります）という考えかたがありません．ある関係を判断するのはあくまでも「真」か「偽」になっています．BASICのプログラミングの際にも参考にすべきでしょう．

なおイコール（＝）の特別な使いかたとしてA＝8などのように，変数に数値を代入する命令があります．

●**文字式**

文字式（文字列式）で使う演算子はただひとつ，「＋」です．文字列（つづり：ストリングス／strings）を結合します．

なお演算子を使わない「"123"」なども式に扱うことがあります．

※600／2とタイプしてリターンキーを押すと，FM-77はこれをプログラムと解釈する．ためしにLISTをとってみるとそれがハッキリする

コンピュータとメモリ

＊メモリには番地が割当てられている

コンピュータは電気（電子回路）の動作で計算をしたり判断をすることは再三述べました．もうひとつ，忘れていけないのは情報を蓄えておくことができることです．これがメモリ（memory：記憶素子）です．人間は自分の都合で憶えたり忘れたり，あるいは外的な条件で忘れたりします．しかしFM-77のメモリは外的な条件で忘れる（つまりスイッチOFF）ことはあっても，それ以外の場合，使う人が指示すればそれに従い，つぎの指示までおなじ数値などを記憶します．

FM-77には6809というCPU（中央演算装置）が使われていますが，このCPUが直接管理できるメモリの数は2^{16}個，つまり65536個分です．このメモリには0番地から始まって65535番地までの，いわばメモリの区画があります．16進数でいえば0000からFFFFまで番地が割当てられています．ただしいま話した1個分というのは1ビットではなく，8ビット分です．つまりメモリの単位番地は8ビット分をひとまとまりとして管理します．

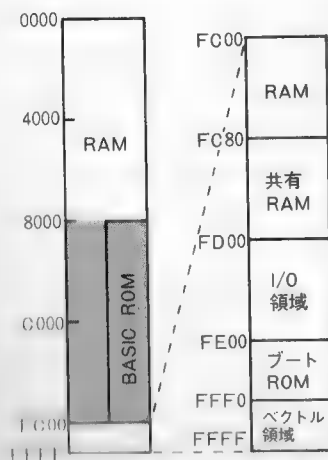
＊＊ユーザーが自由に使えるメモリの量を知る方法

▲メモリマップの例

この8ビットのことを1バイト（byte）といいます．FM-77をONにしたとき

25933 Bytes Free

といったメッセージが出るのは“あなたが自由に使えるのは25933バイト分です”ということを知らせているわけです．6809は65536バイトものメモリを持っているのに，どうしてあなたが使えるメモリが半分以下になってしまうのでしょうか？

それを知るためにも，マニュアルに掲載されたメモリマップを見てください．マップは16進数で書かれており，それぞれの番地がどう使われているのか，あるいは自由に使うことができるメモリの番地の範囲などを教えてくれます．FM-77はメインCPUとサブCPUを持っていますが，一般に私たちが使えるのはメインCPUの，番地でいえば若いほうであることがわかります．また使えるメモリが30キロバイトかそれ以下になる理由はBASIC ROMがRAMにメモリされるからということも分ります．なお，FM-LogoやOS-9などを使うときはFM-77の押ボタンスイッチを操作して，いったんメインメモリをすべて解放して

しまいます．

なお，BASICを使っているときは普通，コンピュータのメモリ…特にメモリ番地は気にしなくてもすみます．BASIC自身がメモリ操作を管理してくれるからです．BASICを使っていて，メモリが直接関係するのはまず使える量があとどのくらいなのか，ということと，使う量をいくつに設定するか，ということです．

＊＊＊メモリの使いわけでコンピュータは効率よく運用できる

さきほどメモリがどれだけ使えるのかをFM-77がまず知らせてくれた例を示しました．ためしに

```
PRINT ("X") ↵
```

としてください．FM-77は

```
300
```

と知らせてくれます．これはFM-77が文字を扱うメモリとして300バイト分を用意してくれている，ということを知らせているのです．

ためしに文字領域を1バイト分だけに設定しましょう．

```
CLEAR 1 ↵
```

そして

```
PRINT FRE("X") ↵
```

とすると，"Out Of String Space"，もう文字用のメモリはありません，というわけです．そのかわりこの状態で

```
PRINT FRE(0) ↵
```

とすると，こんどはスイッチONのときよりも大きな数値が画面に出てくるはずです．このようにF-BASICでは，メモリを文字用と数値用とに分けて設定できますし，それぞれがあとどのくらい残っているかを知ることもできます．

もうひとつ，これはあたりまえのことですが，FM-77が扱う情報が多ければ，使うメモリもたくさん必要になります．たとえば数値がいくつであったか，を記録させるために，もし，とてつもなく大きい数値を正確にメモリに入れるとすると，メモリも当然たくさん必要になります．1,235,678,991,807といった数を扱おうと思ったらFM-77をどう対応させればよいのでしょうか？

あらかじめ倍精度型の宣言（DEFDBL文）をするか，＃記号を使えばよいことになります．しかしすべての変数をこの型で扱うと，当然ですがメモリのムダ使いになるので，じょうずに使い分けましょう．

※FREはメモリの残量を知るための関数．引数（カッコの中に入れる文字や数値）は，文字変数領域を知るときは"X"や"A"など，数値変数領域を知るときは0や1などを使う（引数はダミーなので，とくに制限がない）

キーボードからの入力，いろいろ

＊いちばんよく使うのはINPUT文

キーボードからの入力はコンピュータのもっとも基本的な使いかたのひとつです．多くの場合，入力として求められるのは変数ですが，そうでないものもあります．いずれにしてもキーボードからの入力にはいろいろなスタイルがありますので，それをマスターしましょう．INPUT文が受付けるのは数値変数と文字変数の両方です．サンプルプログラムは3つの例で数値の入力を求めています．行番号120はオーソドックスなもので，プロンプト（前おき，さいそく）の文章に続いてセミコロンをつけ，続いて変数Aを求めています．結果はその右のように，プロンプトのあとに“？”が出て入力をうながします．もしセミコロンではなく，カンマ（，）を使うと，“？”は出ません．つぎの変数Bはプリント文とINPUT文の組合せで，結果的には行番号120とおなじようなものになります．もうひとつはプリント文でプロンプトを作り，行を変えてINPUTさせるものです．

※INPUT A, B, C, ……という書きかたも可能．ただし，入力もたとえば1，2，3，などと，文に合った数だけの入力を“，”を入れて続けたあとにリターンキーを押すので，使いにくい

＊＊じょうずに使えば光ってくるINKEY$

INKEY＄は特別な使いかたになります．リターンキーを押さなくてもよいが，押されたキーのさいしょのひとつだけを受付けること，キーが押されないと空白の入力と見なして，次の行番号へと実行が移されることです．サンプルプログラムを見てください．行番号130で「A＄はINKEY＄である」とプログラムされていますが，この行が実行さ

```
100 '********** INPUT **********
110 CLS
120 INPUT"A ﾊ ｲｸﾂﾃﾞｽｶ ";A
130 PRINT"B ﾊ ｲｸﾂﾃﾞｽｶ ";:INPUT B
140 PRINT"C ﾊ ｲｸﾂﾃﾞｽｶ"
150 INPUT C
200 SUM=A+B+C:PRINT"ｺﾞｳｹｲ";SUM
```

```
A ﾊ ｲｸﾂﾃﾞｽｶ ? 120
B ﾊ ｲｸﾂﾃﾞｽｶ ? 55
C ﾊ ｲｸﾂﾃﾞｽｶ
? 201
ｺﾞｳｹｲ 376
```

INPUT文▶

```
100 '******** INKEY$ **********
110 CLS
120 PRINT "A ﾊ ｲｸﾂﾃﾞｽｶ ?"
130 A$=INKEY$
140 PRINT A$
150 PRINT "B ﾊ ｲｸﾂﾃﾞｽｶ ?"
160 B$=INKEY$:IF B$="" GOTO 160
170 PRINT B$
```

```
A ﾊ ｲｸﾂﾃﾞｽｶ ?

B ﾊ ｲｸﾂﾃﾞｽｶ ?
2
```

◀INKEY$

※空白：なにもないこと，正確にはヌル (null)．「""」といった表現が使われる．「" "」は1文字分の空白になるので，混同しないように注意したい．本書ではヌルを空白と表現している

```
100 '*********** LINE INPUT ************
110 CLS
120 LINE INPUT "ﾄﾞﾝﾅ ﾓｼﾞﾃﾞﾓ ﾄﾞｳｿﾞ ? ";A$
130 PRINT A$
140 PRINT
150 INPUT "ﾄﾞﾝﾅ ﾓｼﾞﾃﾞﾓ ﾄﾞｳｿﾞ ";B$
160 PRINT B$

ﾄﾞﾝﾅ ﾓｼﾞﾃﾞﾓ ﾄﾞｳｿﾞ ? "THIS IS A PEN":"THAT IS A CAT"
"THIS IS A PEN":"THAT IS A CAT"

ﾄﾞﾝﾅ ﾓｼﾞﾃﾞﾓ ﾄﾞｳｿﾞ ? "THIS IS A PEN":"THAT IS A CAT"
?Redo From Start
ﾄﾞﾝﾅ ﾓｼﾞﾃﾞﾓ ﾄﾞｳｿﾞ ? "THIS IS A PEN"
THIS IS A PEN
```

LINE INPUT文▶

れると すぐさまにFM-77は何も入力がなかったと見なして，行番号140のプリント文に実行を移し，空白の文字を出力して，つぎの行番号150に移ります．リストの右側の実行例を見ても分るようにA$を出力するところが1行分，空白になっています．

※リターンキーを押さずに入力できるものにINPUT$があり，指定文字数になると自動的に変数に代入される

INKEY$によって文字の入力を受付けるには，行番号160のように書いて「キー入力の待ち状態」を作らなければいけません．INKEY$の使いかたで多いのは「イエスかノーか」「やめるか続けるか」などのプログラム実行のコントロール時です．リターンキーを押さなくてもよいので，速い動作や操作ミスを防ぐことが可能です．

***すべての入力を受付けるLINE INPUT

INPUT，INKEY$は，それぞれ限定された使いかたであり，限定されているだけに受付けてしまえばそのあとの処理はやさしくなります．その代り，たとえば INPUT A という入力文で“コンニチハ”と入力するとエラーになります(文字列を入力したため)．

もうひとつのLINE INPUT文は原則としてキーボードから入力された文字をすべて受付けるようになっているために，応用の範囲が広がります．上のサンプルを見ても分るように，ダブルコーテーションやコロンなど，ふつうの入力文では文字とは見なさない，特別な意味が持たされている記号も，ちゃんと受けつけます．ここでは，B$はINPUT文になっている（行番号150）ため，エラーになったり，場合によってはキーボードから入力したのに変数には代入されないことがあることが分ります．

どうちがう？IFとWHILE

＊IF文の処理

判断をさせたいときに使うのはもちろんIF文です．

IF 条件 THEN 〔処理〕

IF 条件 GOTO 〔行番号〕

という，2つのスタイルが基本で，この書きかたの場合は条件が偽ならすぐに次の行番号へと実行が移されます．IF文を使うときは，その結果はかならず2つに分岐することを忘れないでください．なお，THEN以下の〔処理〕はほとんどのコマンドやステートメントが使えます．ただし

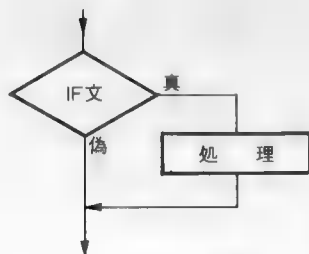
IF A＜＞0 THEN INPUT A

（Aが0でなかったら，Aを入力しなさい）

という使いかたはできません．「INPUT A」を別の行に書いておき，GOTO文でその行へ飛ぶようにします．

IF文にはさらにELSEが使えますが，これについては別に詳しく考えることにしましょう．

▲IF文とプログラムの流れ

サンプルプログラムを見てください．これはX(N)という変数に数値を入れてゆくもので，右がIF文を使った場合です．行番号120で，Nをまず0にセットします．つぎにトラップ（わな）が設けられており，Nが9よりも大きいときは行番号180へ，そうでないときは行番号140へと行きます．ここでNが1つ増やされて，変数に数値が代入され，その後に行番号130へともどるようになります．行番号140でNが10に

```
100 '**** WHILE .... WEND ****
110 '     ｼﾃｲｶｲｽｳﾉ ﾆｭｳﾘｮｸ
120 N=0:CLS
130 WHILE N <10
140    N=N+1
150    LOCATE 0,0:INPUT X(N)
160 WEND
170 '
180 PRINT "N=";N
190 '
200 FOR I=1 TO N
210   PRINT X(I);
220 NEXT
230 END
```

```
100 '***** IF ... THEN *****
110 '     ｼﾃｲｶｲｽｳﾉ ﾆｭｳﾘｮｸ
120 N=0:CLS
130 IF N > 9 THEN 180
140    N=N+1
150    LOCATE 0,0:INPUT X(N)
160 GOTO 130
170 '
180 PRINT "N=";N
190 '
200 FOR I=1 TO N
210   PRINT X(I);
220 NEXT
230 END
```

なり，X (10) からの入力作業を終えると，また行番号の130にもどり，ここでトラップにひっかかって，入力作業を終えるわけです．

**行番号を意識しないで使えるWHILE文

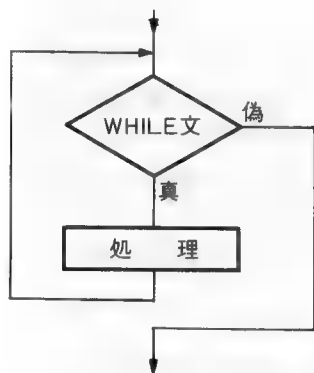
おなじことを，WHILE文で書きました．WHILE文は，そこで与えた条件が真なら，WENDではさんだ文を実行するものです．サンプルプログラムではNが10未満なら行番号140，150をなんどもくりかえします．WEND（ダブルエンド，つまりホワイルエンドのことです）は行番号160に置かれていて，このプログラムの同じ行番号のIF 文における GOTO文とよく似ています．

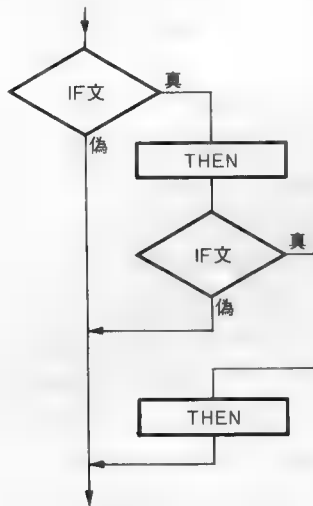
この2つを比べてみてまず気がつくのは，WHILE文とIF文とでは条件の設定がまるで反対になっていることです．IF文では与えた条件の式が真ならTHEN以下，そうでないときは次の行番号を実行するのに対し，WHILE文では与えた条件の式が真なら次の行番号からWENDまでを実行し，偽ならWENDの次からの行番号を実行します．

▲WHILEとプログラムの流れ

WHILE文の副次的な効用としては，サンプルのIF文のように条件を考え，しかもループを作ったときに戻り先を行番号で指定しなければいけないのに対し，WENDは置く位置さえ正しく書くことに注意すれば飛び先や戻り先の行番号を指定しなくてもよいことです． WHILE文は行番号を意識しなくてもよい分，進んだ考えかたともいえるでしょう．プログラムを書く人にとっては行番号そのものは意味を持ちません．意味を持たないどころか，指定行番号をまちがうとエラーになるだけのやっかいもの，という見方からしても，行番号を考えずにすむのはありがたいことです．

なおIF文ではTHEN以下で一度にいろいろな指示を与えることができるのに対し，WHILE文でそれをやろうとすると，場合によってはかなり無理な書きかたになりかねません．慣れないうちはWENDとWHILE間はなるべく簡潔な書きかたにしておくのがよいでしょう．このプログラムではWHILEとWENDの間に2つの行番号をはさんでいますが，行番号130，140，150，160をひとまとめにした多重文（マルチステートメント）にしても，もちろん正しく実行されます．これはサンプルの右側のIF文についても同様で，やはり1行のマルチステートメントにまとめられます．

WHILEはまた，一時停止用にINKEY＄と組合わせてキー入力待ちの状態を作るプログラムによく使われます．

IF～THEN～ELSEの使いかた

＊3つの数の中の最大のものを求める

▲IF～THEN～IF～THEN

A，B，Cの3つの数があるとき，その中の最大のものを見つけるにはどうしたらよいでしょうか？プログラミング上はいろいろな方法が考えられます．まず，もっとも原始的な方法は

A＞Bなら残りのCとAを比べて，A＞CならAが最大

B＞Aなら残りのCとBを比べて，B＞CならBが最大

C＞Aなら残りのBとCを比べて，C＞BならCが最大

という方法で，サンプルのいちばん上がこの手法によるものです．これはまちがいではないのですが，ムダな部分があります．最大値はA，B，Cのどれかなのですから A か B が最大値ではなかったら，自動的にCが最大値になります．したがって，サンプル1の行番号140で，わざわざ比較をする必要はありません．この部分を書き直したのが「IF THEN 2」というサンプルです．

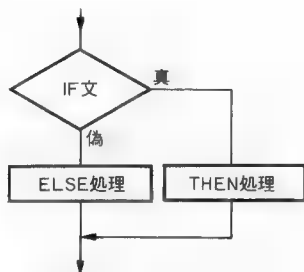
なお，IF～THEN～IF～THENは何重にでも使えますが，ここでは2重に使っています．与えた条件が真ならTHEN以下に進み，偽ならそこで作業は停止してつぎの行に移るのはIF～THENを1回だけ使うのと同じです．

3つの数の中で最大のものを見つける▶

```
100 '=============== IF THEN 1 ===============
110 INPUT"A ";A:INPUT"B ";B:INPUT"C ";C
120 IF A=>B THEN IF A=>C THEN PRINT"MAX=";A:END
130 IF B=>A THEN IF B=>C THEN PRINT"MAX=";B:END
140 IF C=>A THEN IF C=>B THEN PRINT"MAX=";C:END

100 '=============== IF THEN 2 ===============
110 INPUT"A ";A:INPUT"B ";B:INPUT"C ";C
120 IF A=>B THEN IF A=>C THEN PRINT"MAX=";A:END
130 IF B=>A THEN IF B=>C THEN PRINT"MAX=";B:END
140 PRINT"MAX=";C:END

100 '=============== IF THEN 3 ===============
110 INPUT"A ";A:INPUT"B ";B:INPUT"C ";C
120 IF A=>B AND A=>C THEN PRINT"MAX=";A:END
130 IF B=>A AND B=>C THEN PRINT"MAX=";B:END
140 PRINT"MAX=";C:END
```

**IF〜THEN〜ELSEは，ほどほどに

「IF THEN 3」は，IF〜THENを2重に使う代りに論理演算子を使った例です．IF〜THEN〜IF〜THENは真〜真のとき，さいごのTHEN以下を実行しますから，“AND”を使ったのと同じことになるわけです．というわけで，左の3つのサンプルは，すべておなじことをしているわけです．

IF〜THENは，与えた条件が偽のときはすぐに次の行へと移るのに対して，IF〜THEN〜ELSEは与えた条件が真ならTHENで命令したことを，偽ならELSEで命令したことを実行します．

ELSE文の形としては

ELSE GOTO （偽なら指定行番号へ）
ELSE 処理 （偽なら指定の処理をする）
ELSE IF （偽ならもういちど条件を与える）

などがあります．

IF A=10 THEN ○○ ELSE IF A=0 THEN △△ ELSE ××
（もしA=10なら○○を，そうでなくA=0なら△△を，そうでないとき……つまりA=10でもA=0でもなければ××を実行）

といったように，ELSE文も何重にもかさねて使えます．

▲IF〜THEN〜ELSE

下のサンプルはELSE文を使って，A，B，Cの3つの数のうちの最大のものをプリントするものです．左のページの「IF THEN 3」とまったくおなじことをさせていることになります．ELSE文を使うと，ひとつの行番号の中で多くの判断をさせることが可能になるため，プログラムの流れとしては単純になるという利点があります．その反面，いくつもいくつもELSE文が続くと，プログラムが読みにくくなったり，それに伴なって与えた条件のうちのどれとどれを判断させているのかが混乱したりします．1行で長い文を書くと，バグが出たときの修正がめんどうになることもあって，ほどほどにしたほうが扱いやすいといえます．

なおIF文を使うときの注意点は，判断をさせたあとの処理と，処理後にどの行番号を実行するかの，流れをよく見ることです．

※バグ：プログラム上の小さなまちがい

```
100 '=============== IF THEN ELSE 1 ================
110 INPUT"A ";A:INPUT"B ";B:INPUT"C ";C
120 IF A=>B AND A=>C THEN PRINT"MAX=";A ELSE IF B=>A AND B=>C THEN PRINT"MAX=";B
 ELSE PRINT"MAX=";C
```

FOR～NEXT, 使いかたの基本

＊くり返しの回数指定

```
100 '** FOR NEXT & ";" **
110 'SAMPLE 1
120 FOR I=1 TO 10
130   PRINT "A";
140 NEXT I
150 '
160 PRINT"X"
170 END

100 '** FOR NEXT & ";" **
110 'SAMPLE 2
120 FOR I=1 TO 10
130   PRINT "A";
140 NEXT I:PRINT
150 '
160 PRINT"X"
170 END

100 '** FOR NEXT & ";" **
110 'SAMPLE 3
120 FOR I=1 TO 10
130   PRINT "A";
140 NEXT I:PRINTCHR$(13)
150 '
160 PRINT"X"
170 END
```

▲FOR～NEXTとセミコロン

※FOR～NEXT文の制御変数にはよく「I, J, K」などが使われる．これはフォートラン（FORTRAN）というプログラミング言語での使われかたをBASICが受けついでいるためで，一種の習慣になっているが，こだわる必要はない

FOR～NEXTは，この文の間にはさんだ命令を実行するものです．使いかたにはいろいろなバリエーションがありますが，その基本はつぎの2つです．

○単純なくりかえし回数としての使いかた

○FORで使う制御用変数を利用する使いかた

くりかえしの回数指定では，変数になにを使ってもかまいません．たとえば10回のくりかえしなら

FOR I=0 TO 0.9 STEP 0.1

FOR I=9 TO 0 STEP −1

FOR I=101 TO 110

といった使いかたはどれでもおなじことです．FOR～NEXTを使う上で，注意しなければいけないことのひとつは文字出力のときに使うセミコロンと出力される文字列の関係です．左のプログラムは，“A”という文字を10回，続けて出力するために，「“A”;」という形式をとったものです．サンプル1を見てください．行番号160で“X”を出力しますが，サンプル1では

AAAAAAAAAAX

という形で出力されます．FOR～NEXTを10回くりかえして，このループから抜け出しても，セミコロンは“生きて”いますから，カーソルは依然としてさいごのAの文字のあとにあるため，Xという文字はAのうしろにつくわけです．

サンプル2は，NEXT I のあとにPRINT文をつけました．行番号140のPRINT文が実行されるのは，FOR～NEXTのループ動作を終えたあとの1回だけです．ここで空白の文字を出力し，キャリッジリターンされるため

AAAAAAAAAA

X

という形で出力されます．

サンプル3はPRINT文の代りに，キャリッジリターンコードを出力したもので，結果的にはサンプル2とおなじになります．ふつうは，PRINT文を使います．

```
100 '*********** FOR NEXT TIME *************
110 WIDTH 40,20:CLS
120 DEFSNG I:GOSUB 210
130 FOR I=1 TO 10000:NEXT
140 PRINT"I!(ﾀﾝｾｲﾄﾞ)";TIME;"sec"
150 PRINT
160 DEFINT I:GOSUB 210
170 FOR I=1 TO 10000:NEXT
180 PRINT"I%(ｾｲｽｳ)";TIME;"sec"
190 END
200 '
210 TIME$="00:00:00":RETURN '·····ﾀｲﾏ ﾘｾｯﾄ
```

```
I!(ﾀﾝｾｲﾄﾞ) 6 sec

I%(ｾｲｽｳ) 3 sec
```

▲実行時間のちがい

FOR～NEXTに使う変数は，FM-77のスイッチONのままでは単精度になっています（FOR～NEXTに限らず，すべての変数が単精度なのですが）．もちろんこのままで使ってもよいのですが，実行速度を上げたいときで，条件が許されるのなら，整数型の変数を使ったほうが効果的です．上のプログラムはFOR～NEXTを1万回くりかえしたときの，変数の型と実際の処理時間の違いを見るものです．整数型では単精度の約２分の１の時間でFOR～NEXTループを実行することが分ります．整数型を使ったとき，くりかえしは65,536回までできますから，まず十分でしょう．

また，大きい数値から小さい数値へと，逆に使うことが効果的な場合もあります．たとえば大量のデータを配列に読みこむときなどは，読み残りがいくつなのかを画面に知らせておくと，使い手はイライラせずにすみます．

下のプログラムはそのサンプルです．X (I)，Y (I)はI＝360～0と，逆に読みこませていて，どのくらい待つかを行番号130で知らせています．FOR I＝0 TO 360 として，行番号130を (360－I) としても，おなじことですが，前者の方がスマートです．

```
100 '========== FOR NEXT ﾆﾖﾙ ﾊｲﾚﾂﾍﾉ ﾖﾐｺﾐ  SAMPLE ============
110 DEFINT I:DIM X(360),Y(360):PAI=3.14159:WIDTH40,20:CLS
120 FOR I=360 TO 0 STEP -1
130   LOCATE 0,0:PRINT USING"####";I '·········ﾉｺﾘｶｲｽｳ ﾋｮｳｼﾞ
140   X(I)=140*SIN(I*PAI/180):Y(I)=70*COS(I*(PAI/180+.2*PAI))
150 NEXT I
155 '
170 CLS:FOR I=0 TO 360:PSET(300+X(I),100-Y(I)):NEXT I
180 GOTO 180
```

残り回数を知らせる▶

FOR～NEXTと順列，組合せ問題

＊３人がいる時，その並びかたは何通りか？

Ｘ人がいるとき，１列に並ぶとしたらそれは何通りになるでしょうか？数学でいう「順列」の問題です．答はＸ！（Ｘの階乗(かいじょう)）になる，つまり(X)×(X－1)×(X－2)……×(X－(X－1))です．たとえば３人なら３×２×１＝６通りの並びかたになります．

これをコンピュータにさせましょう．例題として３人を考えます．FOR～NEXTは３重のネスティングになります．それぞれは１から３まで変化しますが，あるループの変数が１のとき，他のループの変数が１……つまり同じ数になることはありません．したがってプログラムの中にそれぞれの変数が同じ数になったとき，そのルーチンを飛ばしてやる文を書く必要があります．サンプルプログラムの行番号150，170はそのために書かれています．

行番号170を

IF (I＝J)OR(J＝K)OR(I＝K) THEN 200

としてもよいのです．この書きかたのほうが基本的には正しいのですが，I＝Jのときはすでに行番号150で，行番号210へ飛ばしているので，実際には行番号170の(I＝J)は意味を持たないため，省略しているわけです．

※ネスティング：入れ子構造

```
100 '=============== X ﾆﾝﾉ ﾅﾗﾋﾞｶﾀ =================
110 DEFINT A-Z:N=0:X=3:WIDTH 80,20:CLS
120 '
130 FOR I=1 TO X
140  FOR J=1 TO X
150  IF I=J THEN 210 '...............ｵﾅｼﾞｶｽﾞｦ ﾊｲｼﾞｮ
160    FOR K=1 TO X
170    IF (J=K)OR(I=K) THEN 200 '..ｵﾅｼﾞｶｽﾞｦ ﾊｲｼﾞｮ
180      N=N+1 '......................ｼﾞｭﾝﾚﾂ ｶｳﾝﾄ
190      PRINT USING"##・##・## ";I;J;K;
200    NEXT K
210  NEXT J
220 NEXT I
230 '
240 PRINT:PRINT
250 PRINT USING"◆ ｼﾞｭﾝﾚﾂﾊ ### ﾄｵﾘﾃﾞｽ ◆";N
```

３人の並びかた▶

```
1・ 2・ 3  1・ 3・ 2  2・ 1・ 3  2・ 3・ 1  3・ 1・ 2  3・ 2・ 1

◆ ｼﾞｭﾝﾚﾂﾊ    6 ﾄｵﾘﾃﾞｽ ◆
```

プログラム実行結果▶

```
100 '***** X ノ ナカカラ Y ヲ トリダ゛ス クミアワセ *****
110 '
120 DEFINT A-Z:WIDTH80,20:CLS
130 X=5:Y=3 '........5コノナカカラ 3コヲ トリダ゛ス
140 N=0 '............クミアセスウ カウントヨウノヘンスウ
150 '
160 FOR I=1 TO (X-2)
170   FOR J=I+1 TO (X-1)
180     FOR K=J+1 TO X
190 '
200       N=N+1 '....クミアワセスウ カウント
210       PRINT USING"##・##・##  ";I;J;K;
220 '
230     NEXT K
240   NEXT J
250 NEXT I
260 '
270 PRINT:PRINT
280 PRINT USING"♠ クミアワセハ ### コ デ゛ス ♠";N
```

```
 1・ 2・ 3   1・ 2・ 4   1・ 2・ 5   1・ 3・ 4   1・ 3・ 5   1・ 4・ 5   2・ 3・ 4   2・ 3・ 5
 2・ 4・ 5   3・ 4・ 5

♠ クミアワセハ  10 コ デ゛ス ♠
```

**5枚のカードから3枚を取出す組合せは？

組合せ，という問題もあります．たとえば5枚のカードから3枚を取出す場合，それは何通りあるのだろうか？という考えかたです．これも数学的にはX個あるものからY個を取出す組合せは

(X！)÷[Y！×(X−Y)！]

とされています．5個の中から3個を取出すのなら

(5×4×3×2×1)÷[(3×2×1)×(2×1)]=10

になります．

このプログラムではI=1 TO 3, J=I+1 TO 4, そしてK=J+1 TO 5にしています．J, Kという変数にそれぞれI, Jを使っていることに注目してください．もちろん前のプログラムのような判断文を書くような考えかたもできます．ただし並びかたとはちがって組合せですから，並びかたの（1・2・3）と（3・2・1）は違ったものとして扱うのに対し，組合せではこの両者は同じものになります．5枚のカードから5枚を取出す組合せは1通りしかなく，5枚のカードの並びかたは120通りあります．

乱数をじょうずに使いましょう

＊A以上，B以下の整数を発生させるには

乱数は統計学その他のシミュレーション，身近なところではゲームなどに応用されます．F-BASICの乱数は0以上，1未満なことはよく知っていますが，実際の応用ではこの乱数を整数化して使うことが多いので，INTと組合せます．

INT (RND)

という文で得られる乱数はつねに0です．またAを整数として

INT (RND＊A)

では，0以上，A未満の整数になります．

INT (RND＊B)＋A〔またはINT (A＋RND＊B)〕

はA以上，A＋B未満の乱数が得られます．では《A以上，B以下》の整数をでたらめに発生させるにはどうしたらよいでしょうか？

INT (A＋RND＊ (B－A＋1))

になります．

＊＊毎回，新しい乱数を出す方法

乱数を使いこなすうえで注意しなければいけないのは，ふつうに使ったときは，RUNされるたびにおなじ順序で乱数が発生することです．ですから

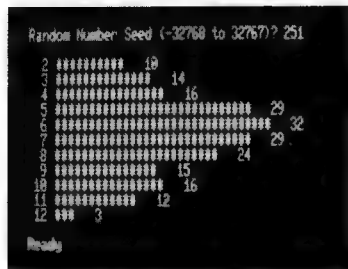
100 PRINT INT (RND＊10)

```
100 '============= ﾗﾝｽｳﾉ ｼﾞｯｹﾝ 1 ==================
110 '
120    DEFINT A-Z:WIDTH 80,20:CLS
130    PRINT"* A ｲｼﾞｮｳ  B ｲｶﾉ ﾗﾝｽｳｦ ﾊｯｾｲｻｾﾏｽ *"
140 INPUT "A ﾊ ｲｸﾂﾃﾞｽｶ ";A
150    INPUT "B ﾊ ｲｸﾂﾃﾞｽｶ ";B
160    IF (B-A)<0 THEN PRINT"NG!!!":GOTO 130
170 '
180    FOR J=1 TO 20:PRINT INT(A+RND*(B-A+1));:NEXT
190 '
200    PRINT:PRINT
210 GOTO 140
```

A以上，B以下の整数を発生▶

```
100 '======== ﾗﾝｽｳﾉ ｼﾞｯｹﾝ 2 ==========
110 IF TIME>30000 THEN TIME$="00:00:00"
120 RANDOMIZE (TIME)
130 PRINT INT(RND*10)
```

実行のたびに新しい乱数を発生するプログラム▶

```
100 '======================= ﾗﾝｽｳﾉ ｼﾞｯｹﾝ 3 ==========================
110 '                  ** ｻｲｺﾛｦ 2ﾂ ｺﾛｶﾞｼﾀﾄｷﾉ "ﾒ" ﾉ ﾃﾞｶﾀ **
120 '=================================================================
130 '
140 DEFINT A-Z:DIM WA(12):WIDTH 80,25:CLS
150 RANDOMIZE:KAISU=200
160 '2ﾂﾉ ﾒﾉ ｺﾞｳｹｲｦ ﾓﾄﾒ ﾊｲﾚﾂﾍﾝｽｳﾆ ｲﾚﾙ
170 FOR I=1 TO KAISU
180  SAIKORO1=INT(1+RND*6):SAIKORO2=INT(1+RND*6)
190  KEI=SAIKORO1+SAIKORO2:WA(KEI)=WA(KEI)+1
200 NEXT I
210 'ｹｯｶﾉ ﾋｮｳｼﾞ
220 FOR I=2 TO 12
230  LOCATE 0,I:PRINT USING"### ";I;:PRINT STRING$(WA(I),"*");"  ";WA(I)
240 NEXT I
250 END
```

▲2個のサイコロを振ったときの目の分布例

というプログラムで，0～9までの整数を発生させたとき，まずRUNをさせて，たとえば「8」が出たとしましょう．もういちどRUNをさせてみてください．やはり「8」が画面に出ます．乱数だからプログラム実行のたびにまったくでたらめの数が発生するだろう，と思ってはいけません．プログラム実行のたびにまったくでたらめの乱数を発生させたかったら，たとえば

```
90 RANDOMIZE(TIME)
100 PRINT INT(RND*10)
```

といったプログラムにしておきます．こうすればランダマイズによって，RUNのたびに新しい乱数が発生します．ただし，いまのプログラムにも落し穴があって，TIME（FM-77の内蔵タイマ．実際には秒数）が32767を越える（スイッチON後，約9時間）と，行番号90でエラーが発生します．ランダマイズを使ったときはサンプルプログラムのように，タイマをリセットする文を書いておけばよいでしょう．

上のプログラムは2個のサイコロを振ったときの目の合計が，どのような分布になるのかのシミュレーションです．それぞれの目が出る確率は数学的に求められますが，画面で見るのも楽しいものです．このプログラムでは2～12までの目に配列変数を使い（WA(I)），FOR～NEXTのループの中でその目が出た回数をカウントさせています．たとえば4が出たら，WA(4)はひとつその数を増やします．ここではサイコロを200回振らせています．

乱数といっても，ある種の「クセ」があることが分るでしょう．

関数とその考えかた

*関数とは

関数 (function) は，数学の世界では変わりうる数と，それに対応する数などを，それぞれ文字で表わした式で説明されます．たとえば$y=x^2-1$といった式はx(独立変数) がいろいろな値をとったときに，y(従属変数) がどう変わるのかを表わす2次関数です．

BASICの関数は広い意味では数学の関数と対応していますが，もっと自由で広い範囲のものを指します．なお，独立変数に相当するものを引数 (argument) といいますが，引数は数値だけでなく，文字列が使えますし，従属変数に相当するものも数値と文字列になります．では関数のいろいろを見てゆきましょう．

**幅の広い，BASICの関数

●数値が引数で，数値が返されるもの

数値関数と呼ばれるものがこれに相当します．三角関数 (SIN，COS，TAN)，逆三角関数 (ATN)，絶対値 (ABS)，指数関数 (EXP)，平方根 (SQR)，正負判定 (SGN)，乱数 (RND) などです．BASIC特有のものとしてはCSNG，CINT，CDBL などの変換関数があります．

●文字列が引数で，数値が返されるもの

ASC (文字列の最初の文字を，ASCIIコードに従った10進数で返す)．例：ASC ("A") の結果は65

INSTR (指定した文字列の中から，見つけたい文字列の位置を探して10進数で返す)

LEN (文字列を与えるとその文字数を返す)，VAL (数値化)

●数値が引数で，文字列が返されるもの

HEX$ (16進数を文字列で返す)，OCT$，CHR$，STR$

●文字列が引数で，文字列が返されるもの

LEFT$，RIGHT$，MID$，STRING$

なお，以上の関数は，引数が1個だけでなく，文字列と数値などの組合せのものがあること，変数だけではなく定数も使えるなどといったことがあるので注意してください．

そのほかの関数としてはキーボード，メモリ，ファイルなどとやりとりをして得られるものがあります．

2

本格的なプログラム作りのために

READ～DATAのいろいろなスタイル

＊データ数をカウントする

READ～DATA文は，プログラムの中にデータがしまえるのでファイル操作なしでいろいろな処理ができます．サンプルのさいしょはデータの数がいくつあるのかをまずカウント（かぞえる）するものです．DATA文の最後に「－9999」を書き，これはダミーのデータとして，実際のデータがいくつかあるかを調べます．応用としては，たとえばデータを配列にしまいたい時，まずデータ数をカウントし，RESTOREしてから，DIM文とFOR～NEXTを組合せて

```
DIM  X(N)
FOR  I=1  TO  N：READ  X(N)：NEXT
```

とすればよいことになります．

また次ページ左のプログラムの行番号200，210のようにすれば目的の数の空白を含む文字列がREADされ，右づめの文字列が作れます．

＊＊文字列とDATA文

文字列をREAD～DATAするときは注意が必要です．DATA文では，個々のデータを区切るためにカンマ（，）を使いますからもとの文字列にカンマがあるものはDATA文として使えません．サンプルの“文字列のREAD”では行番号160のDATAは２つの文字列として読込まれます．行番号170はいくつの文字列として扱われるでしょうか？答は６つです（カンマの合計数＋1）．つまり

①THIS　②空白（ヌル）　③空白（ヌル）　④IS　⑤A　⑥CAT

になります．行番号180や190のように，ダブルコーテーションで囲むと「YES, THIS IS」と「THAT：THAT」というように，それぞれひとつの文字列として読んでくれます．

＊＊＊リストア文，2つの使いかた

なおDATA文はプログラムの行番号のどこに書いてもかまいません．RESTOREの指定がない時は行番号の若い順に読まれてゆきます（“DATA文の位置”のプログラム参照）．

RESTORE文で，その文末に行番号を付けない時は，行番号の若い順に読まれてゆきます．“RESTORE”プログラムはその見本です．サブルーチンでデータを５個ずつ読み，それを画面に出力するようになっています．

まず行番号120で「RESTORE　170」と行番号を指定しているので，DATAは行番号170以降のものが読込みの対象になります．すなわち

1028，2050，3040，5010，10588，25，……

というDATA群になります．このうちの5個が読込まれることになります．

つぎは行番号140で，「RESTORE」と書かれているので

5160，2020，47777，1028，2050，3040，……

が読込まれる対象のDATA群になるというわけで，やはりそのうちの最初の5個のデータが読まれるのです．このプログラムを実際に走らせてみると，いま説明した順序で数字が出力されることが分ります．

グラフィックスのプログラムなどにもDATA文は多く使われますし，RESTORE文をじょうずに使うと効率的なものになります．

◀ ▼READ～DATAのサンプルプログラム

```
100 'READ DATA == ﾃﾞｰﾀｽｳ ｶｳﾝﾄ ==
110 N=0
120 READ DAT
130 IF DAT=-9999 THEN 150
140 N=N+1:GOTO 120
150 PRINT"ﾃﾞｰﾀｽｳ=";N
160 END
170 '
180 DATA 1,2,5,60,500,200
190 DATA -50,62,-55,654
200 DATA -9999 :'READ ｽﾄｯﾌﾟﾖｳ
```

```
100 'READ DATA == ﾓｼﾞﾚﾂﾉ READ ==
110 WIDTH 80,25:CLS
120 READ Y$:IF Y$="ENDEND" THEN END
130   PRINT Y$
140 GOTO 120
150 '
160 DATA THIS IS A PEN,THAT IS A CAT
170 DATA THIS,,,IS,A,CAT
180 DATA "YES,THIS IS"
190 DATA "THAT:THAT"
200 DATA"          THIS IS A PEN"
210 DATA"     OH! THAT IS A CAT"
220 DATA ENDEND
```

```
100 'READ DATA == DATAﾌﾞﾝﾉ ｲﾁ ==
110 DATA 100,500
115 FOR I=1 TO 5
120 READ X:PRINT X,
130 DATA 300,10
140 DATA 2
150 NEXT
```

```
100 'READ DATA == RESTORE ==
110 '
120 RESTORE 170:GOSUB 210
130   PRINT
140 RESTORE:GOSUB 210
150 '
160 DATA 5160,2020,47777
170 DATA 1028,2050,3040,5010
180 DATA 10588,25,26666,21,-500
190 END
200 '
210 FOR I=1 TO 5
220  READ X:PRINT USING"#####,";X
230 NEXT I
240 RETURN
```

PFキーによる割込み操作

＊割込みとは

割込み（interrupt：インタラプト）は，特定条件をあらかじめ設定しておき，その条件になった時，通常の流れから外れて，設定されたルーチンへと飛ぶことをいいます．F-BASICの割込みには通信回線からの情報があった時（ON　COM文），設定時間になった時（ON　TIME文），設定時間間隔になった時（ON　INTERVAL文），設定したPFキーが押された時（ON　KEY文）などがあります．

PFキー（programmable function key）は，ワンタッチで文字列を発生させたり，命令したりできるキーということですがさらに割込み操作にも使える，というわけです．

＊＊2つの入力モードを設定して使いわける

PFキーによる割込み操作の基本は

1 ON　KEY文を書き，どのサブルーチンを呼ぶかをあらかじめ定義しておく

2 割込みを可能にする（KEY　ON），禁止する（KEY　OFF），停止する（STOP）モードを使って，さらに条件づけをする

ものです．流れを一時的に止めて，あるルーチンを呼び，そのサブルーチンが終ったらまた元に戻るのが割込みなので，ON KEY GOSUBという書き方をし，ON　KEY　GOTOとは書きません．

ただし，サンプルプログラムのように，GOSUB文ではあってもリターン文を書く必要のない場合もあります．このプログラムでは，入力ルーチンを2つに設定し，まず行番号210で，どちらの入力ルーチンへ飛ぶかを決めます．もし，入力モードの1を指定したら行番号1000に行きます．ここではON　KEY文で，もしPFキーの7番が押されたら，入力モード2のサブルーチンを呼ぶようにしておき，そののちにPFキーの6（入力モード1）はOFFにします．こうすると画面の下のPFキーの表示部のうち，現在の入力モードは水色に，KEY　ONになっているものは紫色になり，使い手が識別できるようになります．なお，行番号1010～1050の入力処理は特別な書きかたです．

というのは，たとえば入力モード1が定常の流れで，その流れの中でPFキーを押した時，1回だけ入力モード2を実行するのではなく，モード1もモード2も対等に入力できるプログラムになっているから

で，INKEY＄を使い，キー入力の待ち受け状態を作らないとPFキーによる割込みが効かないのです．

このプログラムでは，行番号1020で入力待ち状態を作り，このルーチンを実行中にPF キーが押されると割込みが発生するようになっています．また，行番号1040でINPUT＄を使い，1文字ずつの入力を受付け，リターンキーが押される（ASCIIコードの&H0D）までこれをくりかえして，文字列を受付けているのです．

```
100 '*********************** ｷｰﾉ ﾜﾘｺﾐｿｳｻﾆﾖﾙ ﾆｭｳﾘｮｸ ****************************
110 '
120 '     ｺﾉ ﾌﾟﾛｸﾞﾗﾑﾊ PFKEY 6 ﾃﾞ ﾆｭｳﾘｮｸ1, PFKEY 7 ﾃﾞ ﾆｭｳﾘｮｸ2 ﾉ ﾓｰﾄﾞ
130 '                          PFKEY 10 ﾃﾞ ｵﾜﾘﾆﾅﾘﾏｽ
140 '
150 '*****************************************************************
160 FOR I=1 TO 10:KEY I,"":NEXT '････････････････････PFKEY ｦ ｲﾁﾄﾞ ｶｲｼﾞｮ
170 CLEAR 500:KEY6,"KEY6･･MODE1":KEY7,"KEY7･･MODE2":KEY10,"KEY10･･THE END":WIDTH
 80,20:CONSOLE 0,18,1,0:CLS '････････････････････････KEYﾉ ﾃｲｷﾞ
180 DIM J$(200),F$(200) '･････････････････････････････ﾆｭｳﾘｮｸﾓｰﾄﾞﾆ ﾜｹﾃ ﾊｲﾚﾂｦﾖｳｲ
190 ONKEY(10) GOSUB 3000:KEY(10) ON '･････････････････PFKEY3 ｦ ON
200 Q1$="?  ":Q2$="?? ":J=1:F=1 '････････････････････ﾃﾞｰﾀｽｳ ｶｳﾝﾄﾖｳ
210 INPUT"1･･･INPUT MODE1    2･･･INPUT MODE2  ﾄﾞﾁﾗ";MENU:ON MENU GOTO 1000,2000
220 '-------------------------- PFKEY 1 ------------------------------------
1000 CLS:ONKEY(7) GOSUB 2000:KEY(7) ON:KEY(6) OFF
1010 PRINT Q1$; '･･･････････････････････････････････････ﾌﾟﾛﾝﾌﾟﾄ
1020 A$=INKEY$:IF A$="" THEN 1020
1030 PRINT A$;:J$=A$ '･･････････････････････････････････INKEY$ｦ ﾋｮｳｼﾞ
1040 X$=INPUT$(1):IF HEX$(ASC(X$))="D" THEN 1050 ELSE PRINT X$;:J$=J$+X$:GOTO 10
40 '･････････････････････････････････････････････････････ﾓｼﾞﾉﾆｭｳﾘｮｸ
1050 PRINT:J$(J)=J$:J=J+1:IF J>200 THEN 3000 '････････ﾊｲﾚﾂﾆ ﾀﾞｲﾆｭｳ
1060 GOTO  1010 '･･････････････････････････････････････ﾂｷﾞﾉﾆｭｳﾘｮｸ
1070 '-------------------------- PFKEY 2 ------------------------------------
2000 CLS:ONKEY(6) GOSUB 1000:KEY(6) ON:KEY(7) OFF
2010 PRINT Q2$;
2020 A$=INKEY$:IF A$="" THEN 2020
2030 PRINT A$;:F$=A$
2040 X$=INPUT$(1):IF HEX$(ASC(X$))="D" THEN 2050 ELSE PRINT X$;:F$=F$+X$:GOTO 20
40
2050 PRINT:F$(F)=F$:F=F+1:IF F>200 THEN 3000
2060 GOTO 2010
2070 '-------------------------- PFKEY 3 ------------------------------------
3000 WIDTH 80,25:CONSOLE 0,25,0,0:CLS
3010 FOR I=1 TO J:PRINT USING"&                         &";J$(I);:NEXT
3020   PRINT:PRINT
3030 FOR I=1 TO F:PRINT USING"&                         &";F$(I);:NEXT
3040 END
```

効果倍増！文字列の操作

＊左右中央に文字を出力

メニュー画面や，ゲームの説明などで文字数に関係なくその文字列を左右中央に出力すると，きれいで見やすいものになります．たとえばある文字列が20文字で，1行が40桁の画面なら空白(10)＋文字列(20)で左右中心になり，ある文字列が4文字なら空白(18)＋文字列(4)で左右の中心に出力されます．これを実現するために「TAB」を使ってもよいのですが，いちいち文字数を計算しなければいけません．

このようなときはサンプルプログラムのようにすると，自動的にセンター合わせの文字出力になります．1行の文字数を「LEN」関数を使ってかぞえ，空白数としてSPACE＄を使うものです．このような書きかた以外に，たとえば A＄＝“THE COMPUTER” としてセンター合わせのサブルーチンを呼ぶ，といった方法があります．

ここではDATA文を使ってX＄に代入させていますが，1行をあける手法としてカンマを使っている（行番号200，230）点にも注目してください．

＊＊PRINT USING文では不可能な右づめ出力

文字列や数値を出力するとき，一定のフォーマットが確保される点ではとても便利なものがPRINT USING文です．ところが数値変数は右づめ出力（PRINT USING“＃＃＃＃”）されるのに対し，文字変数は左づめ出力（PRINT USING“&　&”）されてしまいます．たとえば

A$＝"WHITE"
PRINT USING "&□□□□□□□&" ;A$
では
WHITE□□□（□は空白分）

になるのです．数値変数と同じように

□□□WHITE

にするにはどうしたらよいのでしょうか．これも，文字数をかぞえるLEN関数と，空白の文字列を作るSPACE＄と組合わせて実現できます．いま指定の文字幅をTRIMとしましょう．これはPRINT USINGの“&　&”とおなじ働きをするものです．たとえばいま出力桁数を15としましょう．つまりTRIM＝15です．

```
100 '===== ﾓｼﾞﾚﾂﾉ ｿｳｻ **ｾﾝﾀｰ ｱﾜｾ =====
110   INPUT"1ｷﾞｮｳﾊ ﾅﾝﾓｼﾞ(40,80) ";W
120   WIDTH W,20:CLS
130 READ X$:IF X$="ENDEND" THEN END
140  IF LEN(X$)>W THEN170
150  NULL$=SPACE$(INT(W-LEN(X$))/2)
160  X$=NULL$+X$
170  PRINT X$
180 GOTO 130
190 '
200 DATA ******************************,
210 DATA THE COMPUTER GAMES
220 DATA THAT
230 DATA DEVELOPED BY MR.FM,
240 DATA ******************************
250 DATA ENDEND
```

```
******************************

     THE COMPUTER GAMES
            THAT
     DEVELOPED BY MR.FM

******************************
```

▲文字列を指定幅の中央に出力する

▲実行例

```
100 '========== ﾓｼﾞﾚﾂﾉ ｿｳｻ **ﾐｷﾞｻﾞﾒ =============
110 INPUT"ﾅﾝﾓｼﾞﾆ ｶﾘｺﾐﾏｽｶ  ";TRIM
120  INPUT"ｻｻﾞﾘｦ ﾄﾞｳｿﾞ ";X$
130  IF LEN(X$)>TRIM THEN X$=LEFT$(X$,TRIM):GOTO 170
140  '
150  NULL$=SPACE$(TRIM-LEN(X$))
160  X$=NULL$+X$
170  PRINT"1234567890123456789012345" 'ｶﾘｺﾐﾁｪｯｸ ﾖｳ
180  PRINT X$:PRINT
190 GOTO 110
```

```
ﾅﾝﾓｼﾞﾆ ｶﾘｺﾐﾏｽｶ  ? 15
ｻｻﾞﾘｦ ﾄﾞｳｿﾞ ? ｺﾚﾊ ﾐｷﾞｻﾞﾒ
1234567890123456789012345
     ｺﾚﾊ ﾐｷﾞｻﾞﾒ
```

▲文字列の右づめ出力

▲実行例

入力した文字変数から LEN（文字変数）をもとめる……たとえば，X$=“コンニチハ” なら文字数は 5 になります．

空白文字列＝SPACE$（TRIM－5）＝10になります．そして

文字変数＝SPACE$（10）＋（もとの文字変数）

にすれば「空白10個＋コンニチハ」が新しい文字列になり

□□□□□□□□□□コンニチハ

という形で出力されるわけです．

サンプルプログラムは出力桁数を越える長さの文字列については空白を作る必要がない代りに，LEFT$ を使って指定桁数に切り捨てています（行番号130）．なお行番号170は実行結果を確認するためのもので本質的には不要な文です．

文字列の中から目的のものを探す

＊逆のつづりを作る

ここでひとつ，文字の遊びを．入力文字列を逆のつづりにしてみましょう．使う関数はMID＄です．MID＄は指定の位置から指定の文字数を取出す働きをします．入力した文字列の文字数をかぞえたら右はしからひとつずつ文字を取り出し（行番号140），新しい文字列にそれを加えてゆくことで，逆のつづりが完成します．

FOR～NEXTで，「L　TO　1」として，逆の操作にしていること，またGOTO110と，入力ルーチンへ戻しています（行番号170）が，その前にいま作ったつづりを一度空白の文字列（ヌルストリング）にしていることに注意してください．

＊＊文字列の検索

つぎは有用な「INSTR」……インストリングです．これは文字列を指定し，探したい文字列を指定すると，その結果を数値で返してくるものです．いくつかの例を見ましょう．

A$＝〝トウキョウ″：PRINT　INSTR（A$，〝キ″）

では，結果は「３」になります．つまりA＄の中の，左から数えて３番目にキがあるということを知らせてくれました．

A$＝〝トウキョウ″：PRINT　INSTR（A$，〝ヨ″）

は，結果は「０」になります．つまりA＄に指定の文字は含まれていないことを知らせてくれます．

A$＝〝トウキョウ″：PRINT　INSTR（A$，〝キウ″）

も結果は0になります．「キウ」という文字列は含まれていないとコン

```
100 '========= ﾓｼﾞﾚﾂﾉ ｿｳｻ **ｷﾞｬｸﾂﾂﾞﾘ ===========
110 INPUT"ﾂﾂﾞﾘｦ ﾄﾞｳｿﾞ ";NORMAL$
120   L=LEN(NORMAL$):PRINT"length=";L
130   FOR I=L TO 1 STEP -1
140    X$=MID$(NORMAL$,I,1):REVERSE$=REVERSE$+X$
150    PRINT X$,REVERSE$
160   NEXT I
170 REVERDE$="":GOTO 110
```

▲逆のつづりを作るプログラム

```
ﾂﾂﾞﾘｦ ﾄﾞｳｿﾞ ? SUPERMAN
length= 8
N              N
A              NA
M              NAM
R              NAMR
E              NAMRE
P              NAMREP
U              NAMREPU
S              NAMREPUS
```

▲実行例

```
100 '*******    モシ゛レツノ ケンサク <<INSTR>>    ***********
110 DIM X$(10)
120 FOR I=1 TO 10:READ X$(I):NEXT
130 '
140 INPUT "ミツケタイ ナマエ";KY$ '.......キーワート゛ノ ニュウリョク
150 FOR I=1 TO 10
160    N=INSTR(X$(I),KY$)
170       PRINT"I=";I,"N=";N '.....INSTR カンスウノ ナカミ
180    IF N THEN PRINT X$(I) '....シ゛ョウケンニ アウモノノ ヒョウシ゛
190 NEXT
200 GOTO 140
210 'モシ゛レツ テ゛ータ
220 DATA 1ヤマダ゛ キョウコ,2ツカモト シンイチ,3ヤマモト タカシ,4ヤマシタ キョウイチ
230 DATA 5オカモト ヨウコ,6オカヤマ シンシ゛,7タマモト タケシ,8ヤマダ゛ ヨウイチ
240 DATA 9エモト ショウイチ,10オイカワ ミネコ
```

▲インストリングを使って，目的の文字列を探す

```
ミツケタイ ナマエ? ヤマ
I= 1          N= 2
1ヤマダ゛ キョウコ
I= 2          N= 0
I= 3          N= 2
3ヤマモト タカシ
I= 4          N= 2
4ヤマシタ キョウイチ
I= 5          N= 0
I= 6          N= 4
6オカヤマ シンシ゛
I= 7          N= 0
I= 8          N= 2
8ヤマダ゛ ヨウイチ
I= 9          N= 0
I= 10         N= 0
```

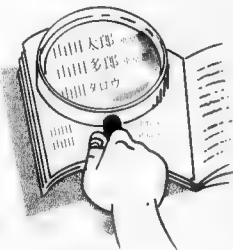
▲実行例

ピュータは知らせてくれます．「キ」も「ウ」も，1文字ならA＄にはありますが，「キウ」はありません，というわけです．

インストリングはこのように非常にすぐれた関数です．たとえば住所録や電話帳から目的のものを探し出すときに，名前の中に「ア」が含まれている人（結果が0でなければ，名前にアが含まれている，という判断文を書けばよい），「ア」から始まる人（結果が1なら，アから始まる），「ア」で終る人（文字列の長さ（文字数）を求め，結果がその数値とおなじならアで終っている）などを見つけることができます．

また同姓同名なら，さらに住所から目的の人を探す（たとえば東京の○○さん）といったように，判断条件を複合化すれば数多くのデータをふるいにかけることができるのです．

上のサンプルでは10人分の姓名を配列に読みこみ，見つけたい名前をKY＄というキーワードにして，それが10人の姓名の中にあるのかどうかを検索させています．INSTRがどういう数値を返してくるかを知るために行番号170で姓名の番号と，INSTRの結果を出力させてから，目的のものを出力させています（行番号180）．

IF　N　という判断文は「Nが0＝偽でないとき」，つまりキーワードが含まれていれば真になるので，その時は姓名をプリントすることになります．INSTRは検索だけでなく，たとえば「しりとり」などの言葉遊びその他にも活用できるでしょう．

多くの文字列の中から
目的のものを探す

解けない？暗号文を作る

＊暗号文の簡単な例

他人には読まれたくない文章は暗号化されます．スパイ活動や戦争などでは欠かせない情報伝達ですし，ポーの小説などにも暗号を扱ったものがあります．暗号は一般に「ある法則」をもとにして元の文を変え，解読はその逆の操作をすることになっています．コンピュータを使ったもっともやさしい暗号文は，英文をカタカナに変えてしまうといったものでしょう．“A”はASCIIコードで65，“ア”はおなじく177です．そこで“A”が入力されたら＋112のコードにすればA→ア，B→イ……といったように変換されます．アルファベットのほうが文字数が少ないのですべてカタカナに変ります．つまり

APPLE……アタタシオ

というわけです．

＊＊キーをファイルにしまう方法

さて正規の暗号文作り，というルールからは外れるかもしれませんが，文字列操作の練習ということで，暗号文を見ただけでは絶対に解説できない？というものを作ってみましょう．元の文の1文字ごとに乱数をあてはめて暗号化します．たとえば元の文章が“ヤマ”としたら乱数を2個使ってみます．最初の乱数が＋3，つぎが＋5だとしたら，ASCIIコードにしたがってヤ→ラ，マ→ヤというように変換します．この乱数は配列にしまっておき，ファイルに書きこんでおきます．暗号化した文章は“ラヤ”になりますが，もともと法則性がないので暗号文を見た人はまず解読できません．解読をするには，まずフロッピーディスケットを入手し，その中のデータのファイル名を見つけ出すと共にプログラムを書かなければいけない，というわけです．つまりデータファイルにある「3，5」という数値をキーにして，暗号化された文章を入力し，ファイルのデータと組合せることで元の文になる，というわけです．

サンプルでは0から29までの乱数を文字数だけ発生させ（行番号260），元の文字に乱数を加えたASCIIコードに変換しています．解読をするときは逆の操作で，データに収められた乱数をキーにして，乱数分だけを引いて戻すことになります．通信の暗号文と，ディスケットのプログラムとデータを使って，初めて解読できます．

```
ｱﾝｺﾞｳﾒｲ? FM77
PRINTER ｦ 1･･････ﾂｶｳ  2･･･････ﾂｶﾜﾅｲ
ﾄﾞﾁﾗﾃﾞｽｶ? 1
ｱﾝｺﾞｳｶ･････1     ｶｲﾄﾞｸ･････2
ﾄﾞﾁﾗ ﾃﾞｽｶ? 1
ﾓｼﾞﾚﾂｦ ﾄﾞｳｿﾞ?  THE ENEMY WENT SOUTHERN ISLAND
hIU/LdQat(`Mk_9pcZe`bb]9RgPYaQ
```

▶暗号化の例

```
100 '================ ﾓｼﾞﾚﾂﾉ ｿｳｻ **ｱﾝｺﾞｳ ====================
110 '           ｷｰﾆﾅﾙ ｽｳｼﾞｦ ﾌｧｲﾙﾆ ｵｻﾒﾙ ﾎｳﾎｳ ｦ ﾂｶｳ
120 '=======================================================
130 F=0:WIDTH 80,20:CLS
140 INPUT"ｱﾝｺﾞｳﾒｲ";AKY$
150 PRINT"PRINTER ｦ 1･･････ﾂｶｳ  2･･･････ﾂｶﾜﾅｲ"
160 INPUT"ﾄﾞﾁﾗﾃﾞｽｶ";P
170 IF P=1 THEN F=1
180 PRINT"ｱﾝｺﾞｳｶ･････1     ｶｲﾄﾞｸ･････2"
190 INPUT "ﾄﾞﾁﾗ ﾃﾞｽｶ";M
200 IF M=1 THEN 220 ELSE IF M=2 THEN 400 ELSE 190
210 '--------------------- ｱﾝｺﾞｳｶ ------------------------
220 LINE INPUT"ﾓｼﾞﾚﾂｦ ﾄﾞｳｿﾞ?  ";X$
230 N=LEN(X$):DIM KY(N)  '･･････････････ﾗﾝｽｳｦ ﾓｼﾞﾉｺｽｳﾀﾞｹ ﾖｳｲ
240 IF TIME>30000 THEN TIME$="00:00:00"
250 RANDOMIZE TIME
260 FOR I=1 TO N:KY(I)=INT(RND*30):NEXT
270 FOR I=1 TO N
280   HENKAN$=MID$(X$,I,1) '･･････････････････1ﾓｼﾞ ｽﾞﾂ ﾄﾘﾀﾞｽ
290     HENKAN$=CHR$(ASC(HENKAN$)+KY(I)) '･･･････ｱﾝｺﾞｳｶ
300   ANGO$=ANGO$+HENKAN$ '･･････････････････････････ﾓｼﾞﾚﾂﾆｽﾙ
310 NEXT I
320 PRINT ANGO$:IF F=1 THEN LPRINT ANGO$
330 'ｷｰﾆﾅﾙ ﾊｲﾚﾂｦ ﾌｧｲﾙﾆ ｵｻﾒﾙ
340 OPEN"O",#1,"0:"+AKY$
350 PRINT #1,N
360 FOR I=1 TO N:PRINT #1,KY(I):NEXT
370 CLOSE:END
380 '---------------------- ｶｲﾄﾞｸ -------------------------
390 'ｷｰﾆﾅﾙ ﾊｲﾚﾂｦ ﾌｧｲﾙｶﾗ ﾄﾘﾀﾞｽ
400 OPEN"I",#1,"0:"+AKY$
410 INPUT#1,N '････････････････････････････ﾓｼﾞｽｳｦ ﾏｽﾞ ﾄﾘﾀﾞｽ
420 DIM KY(N)
430 FOR I=1 TO N:INPUT#1,KY(I):NEXT '････ｷｰﾆﾅﾙ ｽｳﾁｦ ﾄﾘﾀﾞｽ
440 CLOSE
450 LINE INPUT"ｱﾝｺﾞｳｦ ﾄﾞｳｿﾞ?  ";ANGO$
460 FOR I=1 TO N
470   KAI$=MID$(ANGO$,I,1)
480    KAI$=CHR$(ASC(KAI$)-KY(I)) '･･････････････ｶｲﾄﾞｸ
490   KOTAE$=KOTAE$+KAI$'･･････････････････････ﾓｼﾞﾚﾂﾆ ｽﾙ
500 NEXT I
510 PRINT KOTAE$:IF F=1 THEN LPRINT KOTAE$
520 END
```

配列をトランプで考えてみる－1

＊ふつうの変数，配列変数

私たちが使う変数のひとつに配列変数があります．配列変数もアルファベットで始まる変数名をつけなければいけないのですが，普通の変数と決定的に違うのは添字(そえじ)（サブスクリプト：subscript）を使うことと，コンピュータに，あらかじめこの配列変数を使うことを伝える（配列宣言）必要があることです．

普通の変数と配列変数は本質的に異なるものでしょうか？

答はNOです．本質的に異なる点はひとつもありません．しかし場合によっては便利に使えるものです．一般に，配列変数は同じ性質あるいは同じ項目でまとめることができるものを対象として使います．さらにいえば同じ性質なのに，たとえば順番だけが違ったり，部分的に違っているものをまとめて扱うときに使います．

＊＊トランプを例に考える

話を具体的に進めてみましょう．トランプを例にします．カードは（ジョーカーを除いて）52枚あります．52枚のカードは4種類に分かれ，それぞれ1から13までの数になっています．この52枚のカードを切り，伏せて山にしたものから1枚だけを取るとき，それが何であるのかを画面に出すことを考えます．カードを“トランプ”というものと考えて“52枚”あると考えれば，たとえば変数は

```
CARD (1, 2, 3,…52)
```

になります．BASICではこの変数をプログラム中で使う前に

```
DIM  CARD (52)
```

という文…配列宣言文を書きます．こうすると，コンピュータ内部でCARDという名前がつけられたメモリを52個分（正しくは53個分です．このことは後述します）用意してしまいます．この変数は数値変数ですから，メモリに代入できるのは数値になります．

たとえばカードの1番CARD（1）に1を代入するというようにです．これでとにかくカードの1枚目から52枚目までがCARDという配列変数として使えることになります．なお，CARD（1）とCARD1とはまったく別の変数になることを理解してください．CARD（1）はカードの中の1番目，ということで，CARD1はCARD1という独立した変数です．添字は1組，1～52ですから1次元の配列といいます．

52枚のカードはそれぞれ13枚が1組で4種類に分れていますからたとえばこうしましょうか.

1～13：クローバー　→CARD(1)～CARD(13)は1～13
14～26：ダイア　→CARD(14)～CARD(26)は1～13
27～39：スペード　→CARD(27)～CARD(39)は1～13
40～52：ハート　→CARD(40)～CARD(52)は1～13

それぞれのカードには1から13までの数値が代入されます．もし乱数で18が出たら，それはCARD(18)を引いたこととして，その変数に代入されている数は「6」ですから「ダイアの6」というように対応づけることができます．CARDという配列変数と，実際のトランプの種類との対応づけはいろいろな方法が考えられますが，ここでは原始的でもっとも分りやすいものにしておきましょう．

***カードを引く

下のプログラムは52枚のカードに1から13の数値を代入して，そのカードの中から1枚を引くものです．行番号150でNは1から52までになります（例：Iが2になったとき，N＝13×2＋Jになる．Jが13ならN＝26＋13＝39)．カードを引くための乱数は1～52の間として，乱数によってスペードからハートまでの分類をしています．参考のために行番号300，310で乱数の値とCARD(N)を画面に出力しています．なお，このプログラムはカードを1枚引き，それを見たらそのカードをまた山に戻してかきまぜ，また1枚を引くのに似ています．

```
100 '**** CARD *****
110 'ｶｰﾄﾞﾆ ｽｳﾁｦ ｲﾚﾙ
120 DIM CARD(52)
130  FOR I=0 TO 3
140    FOR J=1 TO 13
150       N=13*I+J
160       CARD(N)=J
170    NEXT J
180 NEXT I
190 '
```

```
200 'ｶｰﾄﾞｦ ﾋｸ
210 PRINT"1ﾏｲｦ ﾋｷﾏｽ"
220 PRINT"PUSH ANY KEY"
230 IF INKEY$="" THEN 230
240 N=INT(1+RND*52)
250                             MARK$="♣"
260 IF N>=14 AND N<=26 THEN MARK$="♦"
270 IF N>=27 AND N<=39 THEN MARK$="♠"
280            IF N>=40 THEN MARK$="♥"
290 PRINT MARK$;CARD(N);"ｦ ﾋｷﾏｼﾀ"
300 PRINT USING"{ﾗﾝｽｳ=## ";N;
310 PRINT USING"CARD(N)=##}";CARD(N)
320 PRINT
330 GOTO 210
```

配列をトランプで考えてみる――2

＊カードをまぜて，配ることを考える

はじめの例では，52枚のカードに1～13までの数値を4組つけました．このように配列の添字が1組だけのものを1次元の配列といいます．添字は0から使えますから，DIM CARD(52) では，53個の変数が用意されます．しかし CARD(0) というのはふつうは使いにくいので，ムダですがこれは使わずに CARD(1) から使ったほうが整理しやすいものです．

さて，カードを何枚か，相手に配ることを考えましょう．この場合配ったカードは山には残っていませんから，プログラミングの上では《山に残っているか》，《配ってしまったか》を判定させなければいけません．つまり同じカードを配ってはいけないわけです．とりあえず4枚を配ることにしましょうか．カード配りのアルゴリズム(考えかた・プログラムの書きかたやテーマ実現の手法）も，いろいろなものが考えられています．

ここではCARD(1)～CARD(52) に，まず1～52の数値を入れます．そのあとに，2つの乱数を発生させて，配列にしまわれた数値を

```
100 '=============== 4ﾏｲﾉ  ｶｰﾄﾞｸﾊﾞﾘ  ================
110 '=                                              =
120 '================================================
130 '
140 DEFINT A-Z:DIM CARD(52):WIDTH 40,25:CLS
150 FOR I=1 TO 52:CARD(I)=I:NEXT '・・・・・ｽｳﾁﾉ ﾜﾘｱﾃ
160 '------------------- ｶｷﾏｾﾞ -----------------------
170 IF TIME>20000 THEN TIME$="00:00:00"
180 RANDOMIZE TIME
190 FOR KAI=1 TO 300 '・・・・・・・・・・・・・・・・・・300ｶｲ ｶｷﾏｾﾞﾙ
200  R1=INT(1+RND*52):R2=INT(1+RND*52)
210  SWAP CARD(R1),CARD(R2) '・・・・・・・・・・・ｽｳﾁﾉ ｺｳｶﾝ
220 NEXT KAI
230 '---------- ﾄﾗﾝﾌﾟｦ 4ﾏｲ ｽﾞﾂ ﾋｮｳｼﾞｽﾙ ---------------
240 I=1:K=1 '・・・・・・・・・・・・・・・・・・・・・・・・・・ｼｮｷｾｯﾃｲ
250 IF I>52 THEN END '・・・・・・・・・ｶｰﾄﾞﾉｽﾍﾞﾃｦ ﾋｮｳｼﾞｼﾀﾗ ｵﾜﾘ
260 IF K>4 THEN GOSUB 600 '・・・4ﾏｲﾋﾄｸﾐｦ ﾋｮｳｼﾞｼﾀﾗ ﾂｷﾞﾍ
270   GOSUB 350 '・・・・・・・・・・・・・・ﾊｲﾚﾂﾉ ｽｳﾁｦ ﾄﾗﾝﾌﾟﾉ ｶﾀﾁﾆｽﾙ
280   GOSUB 490 '・・・・・・・・・・・・・・ﾃﾞｨｽﾌﾟﾚｲ
290 K=K+1 '・・・・・・・・・・・・・・・・・・・・・1ﾏｲｽﾞﾂ ｼﾞｭﾝﾊﾞﾝﾆ ﾋｮｳｼﾞ
300 I=I+1:GOTO 250
```

4枚ずつカードを配る▶

```
310 '
320 '***************** SUBROUTINES ******************
330 '
340 '------ 1･･･52ﾉ ｶｽﾞｶﾗ ﾄﾗﾝﾌﾟﾉ ﾏｰｸ ﾅﾄﾞｦ  ﾂｸﾙ-----
350 M=INT((CARD(I)-1)/13)        ' 0･･･3 ﾉ ｶｽﾞｦ ﾂｸﾙ
360 MARK$=CHR$(M+232)
370 '    CHR$(232=♠:233=♥:234=♦:235=♣)
380 '------------ 1････12ﾉ ｶｽﾞﾉ ｿｳｻ ---------------
390 NUM=(CARD(I)-1)MOD13+1 '1･･･12 ﾉ ｶｽﾞｦﾂｸﾙ
400 NUM$=STR$(NUM) '････････････ｽｳｼﾞｶﾗ ﾓｼﾞﾍﾉ ﾍﾝｶﾝ
410 IF  NUM=1 THEN NUM$=" A"
420 IF NUM=10 THEN NUM$="10"
430 IF NUM=11 THEN NUM$=" J"
440 IF NUM=12 THEN NUM$=" Q"
450 IF NUM=13 THEN NUM$=" K"
460 RETURN
470 '
480 '---------------- ﾃﾞｨｽﾌﾟﾚｲ---------------------
490 LOCATE K*8,7 :PRINT       "┌───────┐"
500 LOCATE K*8,8 :PRINT USING"|&&     |";NUM$
510 LOCATE K*8,9 :PRINT USING"| &&    |";MARK$
520 LOCATE K*8,10:PRINT       "|       |"
530 LOCATE K*8,11:PRINT       "|       |"
540 LOCATE K*8,12:PRINT       "|       |"
550 LOCATE K*8,13:PRINT USING"|     &&|";MARK$
560 LOCATE K*8,14:PRINT USING"|    && |";NUM$
570 LOCATE K*8,15:PRINT       "└───────┘"
580 RETURN
590 '-------------- ﾂｷﾞｦ ﾐﾙｶﾄﾞｳｶｦ ｷｸ --------------
600 LOCATE 9,20:PRINT"ﾂｷﾞﾉ 4 ｸﾐｦ ﾐﾙﾅﾗ ANY KEY PUSH"
610 WHILE INKEY$="":WEND  '･････KEYﾆｭｳﾘｮｸﾏﾁ
620 LOCATE 9,20:PRINT SPACE$(30); '･･･ﾒｯｾｰｼﾞｦ ｹｽ
630 K=1 '････････････････････････････Kｦ ﾘｾｯﾄｼ 1ﾆﾓﾄﾞｽ
640 RETURN
```

◀プログラムのつづき

※トランプは1→A，12→Qなどのようになるため，このプログラムでも数値を文字列に変換している．

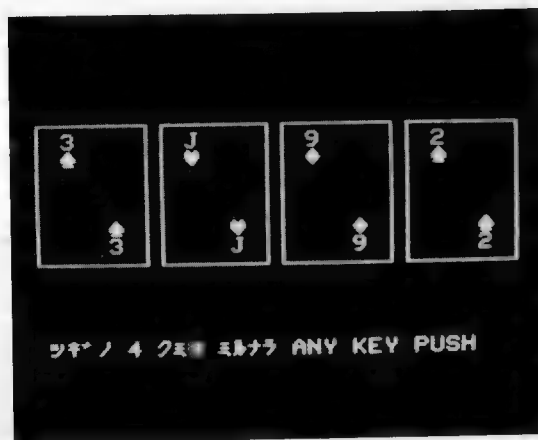
なお，数値は左に1文字分の空白があるので，2ケタの数値を文字列化すると，3個分のスペースになるので，10だけは“10”としている．

交換します（プログラムの行番号190～220）．たとえばR1が5，R2が12という乱数を発生させたとしましょう．さいしょの

CARD(5)＝5　　CARD(12)＝12　は交換されて

CARD(5)＝12　　CARD(12)＝5

になるわけです．ここでは300回のかきまぜ（シャッフル：カード切り）をしています．

こうすることにより，カードを順番に取出すと，中味はバラバラになっていますから，ちょうど本当のトランプのカード切りと同じになるわけです．

◀カードを4枚取出したところ

**1〜52の数をトランプの内容に合わせる

このプログラムでは配列変数の中味は1〜52ですから，やはりマークをどれにするか，1〜12までの数にどう変換するかを考えなければいけません．ここではマークを4種類作るために

1〜13までの数値をすべて0にする，14〜26は1にする……ということのために，INT［(数値－1)÷13］という計算をさせています．たとえば「27」は INT［(27－1)÷13)］＝INT (26÷13)＝2 ですし，「39」も2になるわけです(行番号350)．

また1から12までの数値にするために行番号390で（数値－1）÷13の余り＋1という計算をさせています．たとえば「27」は（27－1）÷13の余り＝0で，それに1を加えるので1になりますし，「40」もおなじように（40－1）÷13＝(39÷13）で余りが0になるわけです．

こうして1〜52の数値がそれぞれのマークとそれぞれ1〜13になったら画面に出力します．このプログラムでは，グラフィック画面は使っていません．したがってキャラクタを出力する位置はすべてロケート文を使っています．このルーチンでは変数は2つ，IとKを使っています．Iは配列の添字をコントロールするために使いますから，I＝I＋1と，1枚のトランプを出力するたびに1ずつ増えます．

KはK＋1と，やはり1ずつ増えますが，1回に4枚のトランプを画面に出したら，また1に戻してやります．同時にK＞4になったら(実際にはK＝5になったら)，次の4枚のカードを見るかどうかを聞くサブルーチンを呼びます．このプログラムはサブルーチン化の必然性はないのですが，リストを見やすくするために使いました．

配列の宣言とERASE文

＊10以下の添字なら宣言なしで使える

配列を使うときに注意することのひとつに，配列変数の宣言があります．DIM文で宣言をするのですが，宣言なしの配列変数はその添字が10と見なされます．もうひとつ，この添字は次元数とは関係しないことも知っておきましょう．宣言をしない状態では

X(10)＝1 →許される　　X(11)＝1 →エラー
X(10, 10, 10)＝1 →許される　　X(10, 10, 11)＝1 →エラー
X(10, 10, 10, 10)＝1 →許される　　X(9, 9, 9, 12)＝1 →エラー

などになります．今の例はそれぞれ1次元，3次元，4次元の配列になっていますが，次元に関係なく，添字が10以下ならよいのです．なお，今の4次元の配列などは用意すべきメモリが多すぎて，実際には使えません（Out Of Memory エラーになる）．

＊＊ERASE文をじょうずに使いこなす

なお，このERASE文をじょうずに使うとメモリが有効に使えます．たとえば，あるプログラムがつぎのような内容をもっているとしましょう．

行番号が1000までに使う配列　X(5000)（それ以降は不要）
行番号が1010から使う配列　Y(5000)

このとき，事前にまとめて DIM X(5000), Y(5000) と書かずに，たとえば 1 DIM X(5000)　1001 ERASE X 1002 DIM Y(5000) といったように書くと，限られたメモリが活かされることになります．

配列変数はDIM文によって必要な個数分のメモリが確保されるので，添字が10以下であっても必らず宣言をする習慣をつけておきたいものです．なお，同一の変数名によるDIM文は1回しか書けません．たとえばある行で DIM X(10) と書き，別の行でも DIM X(18) といった書きかたをすると，2重宣言をしたことになり，エラーメッセージが出ます．

このようなときは，ERASE文によってあらかじめ配列変数を消去しておき，その後にDIM文を書きます．今の例なら，ERASE X としてから DIM X(18) と書くわけです．なおERASE文は配列変数についてだけ適用されます．

多次元の配列もこわくない

＊多次元配列と誤解しやすい部分について

配列は，メモリが許すかぎり何組もの添字が使えます．これらを多次元の配列といい，添字が2個あるものは2次元の，4個あれば4次元の配列になります．たとえば X (A, B) は2次元，X (A, B, C, D) は4次元の配列というわけです．

ところが私たちが普通に使う“n次元”という言葉は，たとえば2次元が平面で，3次元は立体，4次元はさらに時間の要素が加わったもの……という概念があるためか，同じような目で配列を見がちです．配列はそれとはまったく違うものであることをまず理解してください．ある変数を扱うときに，その要素がいくつあるのか，あるいはいくつの要素で扱いたいかによって，多次元化をすればよいのです．

＊＊次元が増えれば変数の種類が減ってくる

配列と次元の話を具体的な例で考えてみましょう．テストの成績ということにします．いま，ここにその個人データがあります．

1学期	国語	80点	数学	65点	英語	70点	社会	95点
2学期	国語	85点	数学	70点	英語	73点	社会	86点
3学期	国語	93点	数学	76点	英語	96点	社会	78点

配列を使わないときは，すべての変数を異なるものにしないといけませんから，たとえばこうします．

```
K1=80:M1=65:E1=70:S1=95:K2=85:M2=70……
```

以下，同じようにして合計で12個の変数を用意するわけです．もし，3学期分の国語の総得点を求めたかったらその変数をKSUMとして，KSUM=K1+K2+K3という式で与えます．

つぎに1次元の配列で処理しましょう．国語の成績はK (I)，算数の成績はM(I)という形にすれば

```
K (1) =80 : M (1) =65 : E (1) =70 : S (1) =95……
```

というようになります．もし国語の3学期分の総得点を見たかったら

```
FOR I=1 TO 3 : KSUM=KSUM+K (I) : NEXT
```

という文を書けばよいことになります．もし，すべての成績の総得点を求めるのなら

```
SUM=KSUM+MSUM+ESUM+SSUM
```

になります．

```
100 '===== 2ｼﾞｹﾞﾝ ﾊｲﾚﾂ 1 =====
110 DIM S(4,3)
120  FOR KAMOKU=1 TO 4
130    FOR GAKKI=1 TO 3
140     READ S(KAMOKU,GAKKI)
150    NEXT GAKKI
160 NEXT KAMOKU
170 '
180 '     1ｶﾞｯｷ 2ｶﾞｯｷ 3ｶﾞｯｷ
190 DATA 80,   85,   93 :'ｺｸｺﾞ
200 DATA 65,   70,   76 :'ｽｳｶﾞｸ
210 DATA 70,   73,   96 :'ｴｲｺﾞ
220 DATA 95,   86,   78 :'ｼｬｶｲ
```

```
100 '======= 2ｼﾞｹﾞﾝ ﾊｲﾚﾂ 2 ========
110 DIM S(4,3)
120  FOR GAKKI=1 TO 3
130    FOR KAMOKU=1 TO 4
140     READ S(KAMOKU,GAKKI)
150    NEXT KAMOKU
160 NEXT GAKKI
170 '
180 '     ｺｸｺﾞ ｽｳｶﾞｸ ｴｲｺﾞ ｼｬｶｲ
190 DATA 80,  65,    70,  95:'1ｶﾞｯｷ
200 DATA 85,  70,    73,  86:'2ｶﾞｯｷ
210 DATA 93,  76,    96,  78:'3ｶﾞｯｷ
```

▲配列へのデータの読みこみ．どちらも結果は同じことになる．

***2次元配列で処理をしてみる

では，もう一歩進めて２次元の配列を使ってすべてをＳという変数で扱うことを考えましょう．このときの要素は何になりますか？ひとつは科目で，ひとつは学期です．科目は４，学期は３ですから

DIM S (4,3)

または

DIM S (3,4)

になります．どちらの書きかたをしてもかまいません．上の例では添字のはじめの要素が科目，２番目が学期になりますし，下の例ではその逆になります．プログラムを書く時に，しっかりと憶えていればよいだけのことです．とりあえず上の例で話を進めましょう．

S (1,1)＝S (はじめの科目，はじめの学期)＝１学期の国語

S (3,3)＝S (３番目の科目，３番目の学期)＝３学期の英語

というように理解をすればよいので，これらのデータ入力にREAD文を使うとすれば別掲のプログラムのようになります．ここではまったく同じ結果を生む２つの例を紹介しています．FOR～NEXTのネスティング（入れ子構造）は，いちばん内側のループから処理されます．左のリストはまず１～３学期の成績を先に読むために，DATA文もそれに対応させていますし，右側のリストでは先に科目が読みこまれるため，やはりDATA文もそれに合わせているわけです．

２次元の配列はよく行列式と対応させて説明されますが，数学の行列演算と配列は一義的には結びつきません．添字がどの要素を意味しているのかをまずしっかりとつかんでおきましょう．

データの並べかえとその応用

*いろいろある並べかえの考えかた

データの並べかえ，それも実用的で時間の速いものをおぼえましょう．人間がデータを並べかえる……たとえば

5 1 3 6 9 2 10

を小さい順に並べるときはどうしますか？ まず片手にノート，片手にエンピツをもち，表の中でいちばん小さいものを見つけ，それをノートに書いて，表の中の数字を消します．残った数字からつぎにまたいちばん小さいものを見つけてノートに書き……という方法をとる人もいるでしょう．人によっては

5 1 3 6　　9 2 10

というようにデータを2組にわけて，それぞれのグループ内で並べかえてやり，その2組から順番に小さいものを見つけ出すかもしれません．データが多くなるほど，この方法は有効です．

※並べかえ＝ソート(sort)

**実用的なもののひとつが“シェルソート”

いずれにしても，人間は眼で見ながらすぐに判断しますが，コンピュータはそうはゆきません．メモリに納められているデータを1個ずつ取出しては比較しないといけないのです．時間がかかる代りに絶対にミスはしませんので，その点はとてもすごいといえます．これから考えるのは“シェルソート”といって，データを半分ずつのグループに分けてやり，それぞれのグループから1対のデータを取出してその大小を比べ，並べかえるものです．特別に速い手法ではないのですが《データは大小さまざまなものがでたらめに並んでいるもの》という前提に立つと，単純な並べかえよりも実際的には速いのです．

プログラムリストを見てください．実際の並べかえは FOR～NEXT のループで行なっています．比べるのはX(I)とX(K)です．たとえばはじめにデータが10個あったとして，Iが1ならKは「データの半分＝5にI(ここでは1)を加えたもの」ですから6，つまりX(1)とX(6)を比べて並べかえをする，というのが基本です．1回分の並べかえが終ったかどうかは行番号220～280の間で判定をし，終りなら行番号190で調べるデータを元の半分にして……ということのくりかえしになります．ここではX(I)に入れる数値を乱数によって求めています(行番号130)．個数はSAMPによってきまります．

データのならべかえ

※WHILE～WENDは，与えられた条件が真のあいだはWHILEとWENDにはさまれた文を実行する

＊＊＊並べかえプログラムの応用と注意点

このプログラムで数値に関係するものなら小さい順に並べかえができますし，もし大きい順に並べかえるのなら行番号250の判断文を書き変えて

IF X(I) < X(K)………

とします．

なお，文字列についてもまったくおなじように並べかえができますが，BASICにおける文字の大小(つづりの順序が先か，あとか）の基準はASCIIのコード表によっていて，カタカナなどもその表の順序で並べかえられます．キャラクタコード表では“ッ”などのほうが“ツ”よりも小さいので（カツコ　カイコ　カッコ）の3つの文字列をこのプログラムを応用して並べかえると（カッコ　カイコ　カツコ）になってしまいます．国語辞典のように並べかえられたものは一般には(カイコ　カツコ　カッコ）ですので注意しましょう．

※カタカナの並べかえは元のデータをカッコ→カツコ，ガッコウ→カツコウなどと，あらかじめ変えておくか，プログラムの上でそれを工夫しておかないといけない．

```
100 '=================== ｼｪﾙｿｰﾄ    ** ｼｮｳ ---> ﾀﾞｲ ** ===================
110 '
120 DEFINT A-Z:SAMP=120:DIM X(SAMP):WIDTH 80,25:CLS
130 FOR I=1 TO SAMP :X(I)=INT(RND(1)*900):NEXT
140 GOSUB 360 '............................ ﾃﾞｰﾀ ﾋｮｳｼﾞ SUB
150 TIME$="00:00:00":'..................... ﾀｲﾏ ｦ ﾘｾｯﾄ
160 N1=SAMP '.............................. ｵｷｶｴ
170 '
180 WHILE N1>1 '........................... ｼﾗﾍﾞﾙ ﾃﾞｰﾀｶﾞ ｱﾙｱｲﾀﾞﾊ ------┐
190 N1=INT(N1/2):N2=SAMP-N1 '.............. ﾃﾞｰﾀｦ 2ﾂﾆﾜｹﾙ                 |
200 LOCATE 0,10:PRINT USING"N1=###  N2=####";N1,N2
210 '
220   FLAG=0
230 FOR I=1 TO N2 '........................ ｲｯﾎﾟｳﾉ ﾃﾞｰﾀｽｳ
240  K=I+N1 '.............................. ｸﾗﾍﾞﾙ ﾃﾞｰﾀﾖｳ
250     IF X(I) > X(K) THEN SWAP X(I),X(K) ELSE GOTO 270
260   FLAG=1
270 NEXT I
280 IF FLAG=1 THEN GOTO 220:'.............. ｿｰﾄｼｭｳﾘｮｳ ﾊﾝﾃｲ                  |
290 WEND '................................. ｺﾉ ﾙｰﾌﾟｦ ｼﾞｯｺｳ .....-----┘
300 '
310 LOCATE 0,12:GOSUB 360 '................ ﾃﾞｰﾀ ﾋｮｳｼﾞ SUB
320 LOCATE 0,21:PRINT TIME$ '.............. ｿｰﾄﾆ ｶｶｯﾀ ｼﾞｶﾝｦ ﾋｮｳｼﾞ
330 '
340 END
350 ' DATA ﾋｮｳｼﾞ SUB
360 FOR I=1 TO SAMP:PRINT USING "#####";X(I);:NEXT
370 RETURN
```

2次元配列を使った表形式の並べかえ

＊それぞれの項目を中心にして並べかえる

まず下のデータを見てください．これはプロ野球のペナントレース中の打撃成績の例です．

名前	打率	打数	安打	打点	勝利打点	本塁打
篠塚	.354	144	51	25	6	4
谷沢	.353	156	55	27	5	11
弘田	.345	139	48	18	4	2
上川	.339	121	41	7	1	6
田尾	.325	169	49	23	5	9

このデータは打率という項目を中心にして成績のよいものから順に並べられていますが，場合によっては「打数が多い順」「打点が多い順」などに並べかえたい時があるものです．もし本塁打順に並べるのなら谷沢(２番目)・田尾(５番目)・上川(４番目)・篠塚(１番目)・弘田(3番目) という名前の順に表が作成されることになります．

＊＊補助の配列を使うと，こんなことが

これを実現するためのひとつの方法として，キー (key：鍵)となる１次元の配列を使ってみます．本塁打という項目に注目した結果

KY (1) = 2 （１位にくるべきものは上から２番目）

KY (2) = 5 （２位にくるべきものは上から５番目）

というようになり，以下 KY(3) =4，KY(4) =1，KY(5) = 3というような結果をKY(I) という配列に入れることができれば

```
FOR I=1 TO 5  （５人分の成績を上から順に）
  FOR J=1 TO 6 （各成績別に）
    PRINT S(KY(I), J);
  NEXT J：PRINT （Jが終ったら改行）
NEXT I
```

という出力をすることで，表形式のならべかえができます．たとえばまず I=1 の時を考えましょう．S(KY(1), J)という配列は，KY(1) = 2 なので，S (2, J) が表の出力のさいしょになります．配列の添字に，配列を使っていることに注目してみれば，なるほど，と納得できることでしょう．

データの大きい順から並べることを考えて「１位のものが上から何

配列の数値から，順位を求める▶プログラム

```
100 '========== DATA ノ キーニナル ハイレツヘノ ジュンイヅケ ===========
110 '
120 DAT=5:DIM X(DAT),KY(DAT)
130 '--------- データノ ヨミコミ
140 FOR I=1 TO DAT:READ X(I):NEXT
150 '--------- キーハイレツヘノ ジュンイヅケ
160 A=1 '......................サイショノデータノ トリダシ
170 WHILE A<=DAT
180 MAX=X(A)
190  JUN=DAT '..................マズ ジュンイヲ サイカイニ シテオク
200     FOR B=1 TO DAT
210       IF MAX>X(B) THEN JUN=JUN-1 '...ジュンイノクリアガリ
220       IF A>B  AND X(A)=X(B) THEN JUN=JUN-1
230     NEXT B
240   KY(JUN)=A '...............キーハイレツヲ ケッテイ
250  A=A+1 '...................ツギノデータヲ トリダス
260 WEND
270 '--------- ケッカノ ディスプレイ
280 FOR I=1 TO DAT:PRINT USING"KY(## )=##";I;KY(I):NEXT
290 '
300 DATA -9,15,60,-90,5
```

```
KY( 1 )= 3
KY( 2 )= 2
KY( 3 )= 5
KY( 4 )= 1
KY( 5 )= 4
```

▲その結果の例

番目か」がわかれば，KY (1)が求められますから，このような表形式の並べかえでは，補助的に使う配列に，どういった手法でこれを実現するかを考えなければいけません．そのためのプログラム例を見てください．

＊＊＊順位づけの考えかた

ここではAという変数とBという変数を使っています．この２つはともにデータの数だけ１，２，３……と変わってゆき，もとの配列の数値のうち，いちばん大きなものが何番目かということを探してゆくものです．MAX（最大値）は１回ごとに配列の値が代入されると共に，さいしょは最下位の順位づけをしておきます．こうしておいて，Bが１からデータ数まで変わり，その中でMAXよりも大きいものがあったら順位がひとつあがるようにしています．

順位を求めたら，KY (I) という配列にそれを代入してゆくということをくり返してゆくものです．左上にその結果のプリントアウト例が出ていますが，KY (1)＝3……３番目のデータがいちばん大きな数値といったような結果になりました．行番号300のデータと比べて正しい順位づけが行なわれていることを確認してください．

なお，この並べかえ判定は大→小への降順ですが，小→大への昇順の時は判定条件を変えてやればよいことになります．また，このよう

```
1-----ダ゛リツ        2------ダ゛スウ          3------ヒットスウ

4-----ダ゛テン        5------ショウリダ゛テン    6------ホームラン

ト゛ノ コウモクデ゛ シ゛ュンイヲ ミマスカ? 5


センシュメイ      ダ゛リツ   ダ゛スウ   ヒットスウ    ダ゛テン  ショウリダ゛テン  ホームラン
-----------------------------------------------------------------------
シノヅ゛カ (G)  354      144      51         25       6           4
タオ      (D)  325      169      49         23       5           9
ヤザ゛ワ   (D)  353      156      55         27       5          11
ヒロタ    (T)  345      139      48         18       4           2
カミカワ   (D)  339      121      41          7       1           6

マタ デ゛ータヲ ミルナラ ANY KEY PUSH
```

勝利打点をキーにして▶
出力したところ

に，データそのものを直接並べかえずに，目印となるものを使って順位づけをする手法はインデックス・ソートといわれます．このインデックスで出力結果を画面に出すには

FOR I=1 TO DAT：PRINT (KY (I))：NEXT

とすればよいわけです．

**** プログラムの実際

では実際に，5人のプロ野球選手の成績を表形式にまとめ，打率から本塁打までの6つの項目のどれからでも，一覧表が出てくるようにしてみましょう．さきほどのインデックスをもとにしたプログラムですからそれほどむずかしくはありません．

ここでは選手の人数分をDAT，1人あたりの項目数をPSという変数にしています．DATA文は選手名と，それぞれの成績を書き，FOR～NEXTで読みこませています（行番号150～190）．この部分をINPUT文にすれば，もちろんいろいろなデータが扱えます．行番号210～240はいわゆるメニュー画面を作成するルーチンです．ここでどの項目を中心にして並べかえるかを求め，何番目の項目かをKという変数に代入しています．

行番号260～360は，Kで指定したデータの順位をKY(I)という配列に入れるもので，前のページと同じものです．こうして出力する順序が決定したので，その結果をディスプレイすればひとつの作業が終ることになります．

なお，“選手名”や“打率”などのいわばタイトルになるものも文字

列変数として扱ったほうが，画面へのディスプレイ時に，出力位置についての苦労は少なくなります．また，その文字列変数がメニュー画面用にも，結果のディスプレイ用にも使えるので本来はそうすべきです．このサンプルプログラムでは並べかえ出力の骨組を知ってもらう目的で変数の種類を少なくしているため，変数を使っていません．実用的に使いこなす時はそれを採り入れるとよいでしょう．

```
100 '================== ﾋｮｳ ｹｲｼｷﾆﾖﾙ ﾃﾞｰﾀﾉ ﾅﾗﾍﾞｶｴ =======================
110 DAT=5:PS=6:DIM S(DAT,PS),KY(DAT),N$(DAT):WIDTH 80,25:CLS
140 '--------- ﾃﾞｰﾀﾉ ﾖﾐｺﾐ
150 FOR I=1 TO DAT:READ N$(I):NEXT  '･･･････ｾﾝｼｭﾒｲ
160 FOR I=1 TO DAT '････････････････････････ｾﾝｼｭﾌﾞﾝｦ ﾏｽﾞ ﾖﾐ
170  FOR J=1 TO PS '････････････････････････ｾｲｾｷ
180  READ S(I,J)
190 NEXT J,I
200 '--------- ｿｰﾄｼﾀｲ ｺｳﾓｸﾉ ﾆｭｳﾘｮｸ
210 PRINT"1-----ﾀﾞﾘﾂ      2------ﾀﾞｽｳ        3------ﾋｯﾄｽｳ":PRINT
220 PRINT"4-----ﾀﾞﾃﾝ      5------ｼｮｳﾘﾀﾞﾃﾝ    6------ﾎｰﾑﾗﾝ":PRINT
230 INPUT"ﾄﾞﾉ ｺｳﾓｸﾃﾞ ｼﾞｭﾝｲｦ ﾐﾏｽｶ";K
240 IF K<1 OR K>6 THEN CLS:GOTO 210
250 '--------- ｷｰﾊｲﾚﾂﾍﾉ ｼﾞｭﾝｲﾂﾞｹ
260 A=1 '･････････････････････････････････････ｻｲｼｮﾉﾃﾞｰﾀﾉ ﾄﾘﾀﾞｼ
270 WHILE A<=DAT'･･････････････････････････････ﾃﾞｰﾀｶﾞ ｱﾙｱｲﾀﾞﾊ
280 MAX=S(A,K) '･･･････････････････････････････ｶﾘﾉ ｻｲﾀﾞｲﾁｦ ｷﾒﾃｵｸ
290  JUN=DAT '･････････････････････････････････ﾏｽﾞ ｼﾞｭﾝｲｦ ｻｲｶｲﾆ ｼﾃｵｸ
300     FOR B=1 TO DAT
310       IF MAX>S(B,K) THEN JUN=JUN-1 '･･････････ｼﾞｭﾝｲﾉ ｸﾘｱｶﾞﾘ
320       IF A>B  AND S(A,K)=S(B,K) THEN JUN=JUN-1 'ｸﾘｱｶﾞﾘ
330     NEXT B
340   KY(JUN)=A '･･･････････････ｷｰﾊｲﾚﾂｦ ｹｯﾃｲ
350  A=A+1 '･･･････････････････ﾂｷﾞﾉﾃﾞｰﾀｦ ﾄﾘﾀﾞｽ
360 WEND
370 '--------- ｹｯｶﾉ ﾃﾞｨｽﾌﾟﾚｲ
380 PRINT:PRINT:PRINT"ｾﾝｼｭﾒｲ      ﾀﾞﾘﾂ    ﾀﾞｽｳ    ﾋｯﾄｽｳ     ﾀﾞﾃﾝ  ｼｮｳﾘﾀﾞﾃﾝ  ﾎｰﾑﾗﾝ"
390 PRINT"-------------------------------------------------------------"
400 FOR I=1 TO DAT
410 PRINT USING"&          &";N$(KY(I));'･･･････････････････････ｾﾝｼｭﾒｲ
420    FOR J=1 TO PS:PRINT USING" ###     ";S(KY(I),J); '･･････ｾｲｾｷ
430 NEXT J:PRINT:NEXT I
500 PRINT:PRINT"ﾏﾀ ﾃﾞｰﾀｦ ﾐﾙﾅﾗ ANY KEY PUSH"
510 IF INKEY$=""THEN 510 ELSE CLS:GOTO 210
600 DATA "ｼﾉﾂﾞｶ (G)","ﾔｻﾞﾜ  (D)","ﾋﾛﾀ   (T)","ｶﾐｶﾜ  (D)","ﾀｵ    (D)"
610 DATA 354,144,51,25,6, 4 :'ｼﾉﾂﾞｶ
620 DATA 353,156,55,27,5,11 :'ﾔｻﾞﾜ
630 DATA 345,139,48,18,4, 2 :'ﾋﾛﾀ
640 DATA 339,121,41, 7,1, 6 :'ｶﾐｶﾜ
650 DATA 325,169,49,23,5, 9 :'ﾀｵ
```

配列をゲームに応用する

＊もっともやさしいゲームを例にしてみる

配列はいろいろな数値計算などに使えると共に，ゲームなどへの応用にも欠かせません．シミュレーションゲームやクイズ形式のもの，またオセロなどのいわゆる“マス目”を使ったものなどに配列変数が活躍します．配列とゲーム感覚を養うために，もっともシンプルなものを手本にして考えてみましょう．図の３×３のマス目の左上がスタート，右下がゴールのいわば区画された野原です．この区画のどこかに爆弾がかくされています．それをよけてゴールに着けば成功，爆弾の上に乗ったら失敗，ということにします．このゲームでは

○爆弾をかくすこと

○移動しながら，各区画へ行くこと

○爆弾の上に乗ったか，ぶじにゴールについたかなどの判定をすること

のアルゴリズムが見つかれば，プログラムを書くことができます．

＊＊爆弾を“かくす”ということの実際

ではこの９つの区画に爆弾をかくしてみましょう．まずＦ(3,3)という配列を用意します．この配列の，スタート地点であるＦ(1,1)とゴールのＦ(3,3)以外のどこかに爆弾をかくせばよいので，乱数によって，１から３までの数値を発生させます．とりあえずひとつはBX，ひとつはBYとしましょうか．BX＝BY＝1，BX＝BY＝3にならない乱数を求めたら，Ｆ(BX，BY)＝１と，この配列変数だけを１にして，そのほかの配列変数はすべて０という値にすることで，この区画のどこか１ヵ所だけが他の区画とは違うものになります．つまり，ここを爆弾のある場所ということにします．

つぎに移動させることを考えます．どの方向に行くかはカーソルキーを使うことにしましょう．右向きの矢印を押したらＸという変数がひとつ増え，左向きならひとつ減り……というようにすると，さいしょはＸ＝１として，１回だけ右向きのカーソルキーを押せばＸ＝Ｘ＋１になるようにしておけば，これを１度押すとＸ＝Ｘ＋１で２になります．上下方向もおなじ考えで処理できます．

```
******** HIDE BOMB ********

 ┌─┬─┬─┐
 │*│?│?│   *カラ スタート
 ├─┼─┼─┤
 │?│?│?│
 ├─┼─┼─┤ カーソルKEY デ゛ イト゛ウ
 │?│?│G│
 └─┴─┴─┘

           G カ゛ ゴ゛ール
```

こうして新しい移動先を求めたら，そこに爆弾があるかないかを判定して，爆弾があったらゲームセットになります．具体的に考えてみましょう．乱数でBX＝２，BY＝１が発生した，とします．するとＦ

```
100 '****************** HIDE BOMB !!!! ***********************
110 '
120          DEFINT A-Z:DIM F(3,3):WIDTH 40,25
130 COLOR 4,0:CLS
140         IF TIME>20000 THEN TIME$="00:00:00"
150 '--------------------- ｼｮｷｾｯﾃｲ -------------------------
160         FOR X=1 TO 3:FOR Y=1 TO 3:F(X,Y)=0:NEXT Y:NEXT X
170 '------------------- ﾊﾞｸﾀﾞﾝｦ ｶｸｽ ------------------------
180                RANDOMIZE TIME
190 BX=INT(1+RND*3):BY=INT(1+RND*3)
200      IF BX=1 AND BY=1 THEN 190
210      IF BX=3 AND BY=3 THEN 190
220        F(BX,BY)=1'ｺｺﾆ ﾊﾞｸﾀﾞﾝｶﾞ ｱﾘﾏｽ
230 '
240 '******************** ｹﾞｰﾑ ｶｲｼ ***************************
300   X=1:Y=1'ｻｲｼｮﾉｲﾁﾉ ｾｯﾃｲ:                                '*
310   GOSUB 740'ｹﾞｰﾑﾊﾞﾝｦ ｶﾞﾒﾝﾆﾀﾞｽ:                           '*
320 GOSUB 390 'ｲﾄﾞｳ ﾆｭｳﾘｮｸ:                               '*
330   GOSUB 520 'ｲﾄﾞｳ:                                    '*
340   GOSUB 530 'ﾊﾞｸﾀﾞﾝｶﾞｱﾙｶ ﾅﾄﾞﾉ ﾊﾝﾃｲ:                      '*
350 GOTO 320:                                             '*
360 '*******************************************************
370 '
380 '------------------- ｲﾄﾞｳ ﾆｭｳﾘｮｸ -------------------------
390 A$=INKEY$:IF A$="" THEN 390
400  IF ASC(A$)=28 THEN X=X+1 'ﾐｷﾞ
410    IF X>3 THEN X=X-1:GOTO 500
420  IF ASC(A$)=29 THEN X=X-1 'ﾋﾀﾞﾘ
430    IF X<1 THEN X=X+1:GOTO 500
440  IF ASC(A$)=30 THEN Y=Y-1 'ｳｴ
450    IF Y<1 THEN Y=Y+1:GOTO 500
460  IF ASC(A$)=31 THEN Y=Y+1 'ｼﾀ
470    IF Y>3 THEN Y=Y-1:GOTO 500
480 RETURN
490 'ｴﾗｰﾖｳﾉ ｵﾄｦ ﾀﾞｽ
500 BEEP1:FOR T=0 TO 200:NEXT:BEEP0:GOTO 390
510 '--------------- ｲﾄﾞｳｻｷﾆ ﾏｰｸｦ ﾀﾞｽ ----------------------
520 LOCATE 9+X*2,8+Y*2:PRINT"*";:RETURN
530 '------------------ ｼｮｳﾊﾟｲｶ ｾｲｺｳｶ -----------------------
540 IF F(X,Y)=1 THEN GOTO 590 'ﾊﾞｸﾀﾞﾝﾆｱﾀﾘ!
550 IF X=3 AND Y=3 THEN 660 'ﾀﾞｲｾｲｺｳ!!!
560 RETURN
570 '
580 '---------------- ﾊﾞｸﾀﾞﾝﾆｱﾀｯﾃ ｼｮｳﾊﾟｲ --------------------
590   LOCATE 10,20:PRINT"BANG!!!!!"
600   FOR I=1 TO 10:PLAY"T250S3L8O3C":PLAY"":NEXT
610   FOR T=0 TO 1000:NEXT
620   FOR C=1 TO 7:COLOR 4,C:CLS:FOR T=0 TO 200:NEXT T:NEXT C
```

※→はASCの28
←はASCの29
↑はASCの30
↓はASCの31

（次ページに続く）

(2,1)＝1になっています．スタート地点はF(X, Y)です．さいしょはX＝1，Y＝1なので，F(1,1)にいます．F(1,1)は当然，0になっていますから，爆弾の上ではありません．右向きのカーソルキーを押しました．X＝X＋1で2になります．Yは1のままなので，F(2,1)になりました．F(2,1)が0ならまた移動の入力をしますが，F(2,1)＝1なのでここは爆弾の上ということで，残念ながら失敗というわけです．

***基本はごく簡単

このゲームの基本はこのようにごく簡単です．ただ，いかにもゲームらしくするためには画面に区画を描いたり，移動させたらその位置に新しくマークを作ることや，爆発したらそれらしい音を加えたりするなど，いわばゲームを盛りあげるアクセサリー的な部分があるためにプログラムリストはやや長い行数になります．実際的なゲームの骨組は行番号300～350のようになっています．カーソルキーを押すたびに＊マークが移動先を示しますが，設定された区画の外に出ないように工夫をすることと，カーソルキーによって発生したXやYの値とLOCATE文との関係，そしてゴールに着いたことの判定方法が分れば，このプログラムをすっかり理解したことになります．

```
630   COLOR 4,0:CLS
640 GOTO 700
650 '--------------------  ﾀﾞｲｾｲｺｳ ----------------------
660   LOCATE 10,20:PRINT"OK OK OK !!!!"
670   PLAY"S2T6005GEGEGE"
680   LOCATE 9+BX*2,8+BY*2:COLOR 2:PRINT"●" 'ｶｸｻﾚﾃｲﾀ ﾊﾞｼｮ
690 '------------------- ｹﾞｰﾑｦ ﾂﾂﾞｹﾙｶ ---------------------
700 LOCATE 10,22:PRINT"REPLAY? Y or N"
710   YN$=INKEY$:IF YN$="" THEN 710
720   IF YN$="Y" OR YN$="y" THEN 130
730   IF YN$="N" OR YN$="n" THEN END ELSE 700
740 '--------------------- ｹﾞｰﾑﾊﾞﾝ -----------------------
750    LOCATE6,5:PRINT"******** HIDE BOMB ********"
760    LOCATE 10, 9:PRINT"┌─┬─┬─┐"
770    LOCATE 10,10:PRINT"│*│?│?│   *ｶﾗ ｽﾀｰﾄ"
780    LOCATE 10,11:PRINT"├─┼─┼─┤"
790    LOCATE 10,12:PRINT"│?│?│?│"
800    LOCATE 10,13:PRINT"├─┼─┼─┤ ｶｰｿﾙKEY ﾃﾞ ｲﾄﾞｳ"
810    LOCATE 10,14:PRINT"│?│?│G│"
820    LOCATE 10,15:PRINT"└─┴─┴─┘"
830    LOCATE 10,16:PRINT"          G ｶﾞ ｺﾞｰﾙ"
840 RETURN
```

3

すっかり分る

ファイルの操作

シーケンシャルファイルの基礎——1

＊ディスクと本体との窓口を開設する

ファイル（file）はとじこみ，といった意味ですが，ここではとりあえずデータが書きこまれた書類と考えればよいでしょう．FM-77で作ることができるデータファイルは2種類あります．

シーケンシャルファイル（sequential file）：連続記録型

ランダムアクセスファイル（random access file）：ページ単位記録型のファイル

どちらのファイルを使うかは，扱うデータの内容などによって異なります．まずシーケンシャルファイルをマスターしましょう．このファイルを使うときに絶対に必要なことは

○ファイルのタイプを指定すること

○データを書きこむのか，読出すのかを指定すること

○コンピュータ側の仕事なのか，ディスク側の仕事なのかを指定すること

です．まず，もっとも簡単なファイルを作ってみましょう．FM-77を起動させたときの，“How many disk files (0-15) ?”のとき，数字を入力しないで，単にリターンキーを押してください．これにより，FM-77の内部ではディスクとのやりとりをするチャンネル（窓口）がひとつだけ設定され，その窓口はふつう1番（＃1）にします．べつに＃1にしなくても，このひとつの窓口の番号を＃5にしても，＃10としてもよいのですが，窓口がひとつですからすなおに＃1としておきましょう．

※プリンタや通信回線，キーボードなどの，入出力装置のすべてをファイルとして考えることもできる．これらのファイルでは，まずどのファイルなのかを指定（例：CAS, COM）してから使う．ここでいうファイルはディスクドライブを使った，データファイルを意味する

＊＊データを書きこまなくても，ファイルは作られる

まずディスクドライブの0番に，書きこむことのできるディスケットを入れてから

```
OPEN "O", #1, "0:TEST
```

とタイプして，リターンキーを押してください．ディスクドライブが動作します．ためしに

```
FILES "0:
```

とタイプし，リターンキーを押すと，0番に入れたディスケットに何が収められているかを知ることができ

```
TEST        1AS1
```

とメッセージが出るでしょう．まだデータは何も書きこんでいなかったはずですが，とにかくこのような処理がされます．さきほどOPEN命令で，FM-77とディスクドライブのやりとりの窓口を開けていますから，いちどこの窓口を閉じます．

CLOSE　#1

とタイプし，リターンキーを押します．では，本当にディスケットにTESTという名前でデータファイルが収められているのでしょうか？ためしにFM-77のスイッチを切り，再度起動させます．そののち

FILES "0：

で，ディスケットの中味を見てください．やはりTESTという名前のデータファイルが存在していることが分ります．さて，これはまったく意味のないものですから，ディスケットから消し去りましょう．

KILL "0：TEST

とタイプしてリターンキーを押すと

Are you sure (Y or N) ?

本当に消してしまっていいのですね？というプロンプトが出ますからYをタイプすれば消し去ることができます．

※ディスクドライブの0番を使うときは番号が省略できる．
OPEN "O", #1, "TEST"やFILESなどはすべて0番を指定しているのとおなじことになる

＊＊＊データはどうしまわれているのか？

いまのようにOPEN文で窓口を開け，CLOSE文で閉めることと，ファイルの番号を指定すること，ディスケットの中味を知るためのFILES命令などが，本体とディスクドライブとの基本的なやりとりになります．さらにデータを書きこむときは

PRINT #1, A

などという文を書きます．この"A"はディスケットにAという文字を書くのではなく"現在，FM-77の内部でAという変数名のもとに記憶されている数値をディスケットに収めなさい"という意味です．また，データを読出すとき

INPUT #1, X

という文を書いたら，現在FM-77とディスクドライブでデータの受け渡しができるもののうち，最初に見つけた数値データをXという変数名のメモリにしまうことを意味します．

なお，ディスケットには「変数名と共にデータが収められている」わけではありません．書きこんだ時の順序にしたがってデータだけが収められていることをよく理解しましょう．

※CLOSEとすると，すべてのファイルが閉じられる

シーケンシャルファイルの基礎――2

***読み，書きの基本**

ではディスケットにデータを書きこみ，それを読出す実験をしてみましょう．まず，A$に文字列を，Xに数値を代入します．その次にこのデータを収めるときのファイル名を何にするかを求めて，それをFLNM$に代入しておきます．準備が終ったらOPEN文で，#1のファイルを指定し，A$，Xの2つの変数の中味をディスケットに収めましょう．

念のため，行番号230でA$とXの変数の中味をFM-77の内部から消し去っておきましょう．本当にディスケットから読出しかどうかをチェックするために，です．さて，ディスケットに収められたものの内容を見たかったら“Y”を入力します．データをFM-77に読ませるために，入力のための文を書きこみます．この時，書きこんだファイル名を指定しなければいけません．ここではFLNM$に代入されている文字列で指定しています．

****ディスケットにはどのような形で収められているのだろう**

さて，ではまずさいしょの文字列を読出しましょう．ここでは変数名にX$を使い文字列を，Aに数値を代入します．注意してほしいのは，ディスケットには文字列と数値だけが単なるデータとして記録されているだけですから，文字列なら文字列変数，数値なら数値変数を使いさえすればよいことです．書きこむ時にはA$に代入されていたものをディスケットに収め，読出す時はX$を使っても，いっさいの不都合はありません．

なお，断るまでもないことですが，わざわざ変数名を変える必要はないので，本番のプログラムは，このサンプルのように書いてはいけません．いわばこれは悪い見本なのですが，収められたデータの本質がどういうものであるのかを知ってほしいために，あえて変えたものなのです．というわけで，ディスケットから読出したデータが画面に出力されます．

*****自由に読出すためには**

このプログラムを一度実行すると，あるファイル名のもとで文字列がひとつ，数値がひとつ，ディスケットに収められます．では，いつでも自由にこのデータを読出すためにはどうしたらよいのでしょう？

```
100 '========== PRINT#,INPUT# SAMPLE 1 =======
110 WIDTH80,20:CLS
120 INPUT"ｽｷﾅ ﾓｼﾞｦ ﾄﾞｳｿﾞ ";A$
130 INPUT"ｽｷﾅ ｽｳﾁｦ ﾄﾞｳｿﾞ ";X
140 INPUT"ﾌｧｲﾙﾒｲﾊ ﾅﾆﾆ ｼﾏｽｶ(8ﾓｼﾞｲﾅｲ) ";FLNM$
150 '
160 '--------------- PRINT # ----------------
170 OPEN "O",#1,"0:"+FLNM$ '･･････ﾄﾞﾗｲﾌﾞ0ｦ ﾂｶｳ
180 PRINT #1,A$ '･･････････････････ｶｷｺﾐ
190 PRINT #1,X '･･････････････････ｶｷｺﾐ
200 CLOSE '････････････････････････ﾏﾄﾞｸﾞﾁｦ ﾄｼﾞﾙ
210 '
220 '-------------- ﾃﾞｰﾀｦ ﾐﾏｽｶ ---------------
230 A$="":X=0 '･･･････････････････ﾍﾝｽｳｦ ｹｽ
240 PRINT:PRINT
250 INPUT"ｲﾏ ｷﾛｸｼﾀﾓﾉｦ ﾄﾘﾀﾞｼﾏｽｶ(Y or N)  ";YN$
260 IF YN$="N" OR YN$="n" THEN PRINT"END":END
270 IF YN$="Y" OR YN$="y" THEN 300 ELSE 250
280 '
290 '--------------- INPUT # ----------------
300 OPEN "I",#1,"0:"+FLNM$ '･･････ﾏﾄﾞｸﾞﾁ ｶｲｾﾂ
310   INPUT #1,X$ '････････････････ﾃﾞｰﾀ ｦ ﾖﾑ
320   INPUT #1,A '････････････････ﾃﾞｰﾀ ｦ ﾖﾑ
330 CLOSE '････････････････････････ﾏﾄﾞｸﾞﾁｦ ﾄｼﾞﾙ
340 '
350 '-------------- ﾃﾞｨｽﾌﾟﾚｲ ----------------
360 PRINT"ｶｷｺﾏﾚﾀ ﾓｼﾞ=";X$
370 PRINT"ｶｷｺﾏﾚﾀ ｽｳﾁ=";A
380 END
```

PRINT＃, INPUT＃の基本プログラム▶

たとえばFLNM＄に“TEST”を使ったとします．また，このプログラムを“SAMPLE”という名前でディスケットにSAVEした，という前提で話を進めます．いちどスイッチを切り

LOAD "SAMPLE"

とすると，このプログラムがロードされます．この時点ではすべての変数は空ですから，まず

FLNM$＝"TEST"

としてリターンキーを押します．次に

RUN 300（あるいはGOTO 300）

とタイプしてリターンキーを押してみてください．このプログラムの行番号300から実行され，ファイルからX＄とAというデータを読出して，画面に出力します．

シーケンシャルファイルの基礎－3

＊配列に入っているデータをファイルに収める

つぎに，数多くのデータが配列にしまわれている場合，それをファイルへ書きこむと共に読出す方法についても考えてみましょう．まず，右のページの上段のリストを見てください．30個のデータをREAD文で読み，それを配列にしまい，さらに“SAMP. X30”というファイル名でディスケットに書きこむプログラムです．データの数がいくつあるのかを，N％という変数を使って処理しています．

データをしまうときにも，まずこのN％を書きこみ，そののちにN％回だけFOR～NEXTをくりかえしています．このプログラムは，いわば書きこみ専用ですから，読出すためには別のプログラムを使うことになります．

※ENDが実行されると，ファイルは自動的にCLOSEされる

＊＊逆操作で読出す

データを読出すプログラムのサンプル1が右ページ中段のものです．ここではファイルをオープンし，まずデータの数がいくつなのかを知るためにN％という変数を使います．これを求めたら，DIM文で配列宣言をしたのち，書きこみプログラムとおなじようにFOR～NEXTを使ってデータをX(I)に代入しています．書きこみプログラムと読出しプログラムは1対1の対応をしている，ごくオーソドックスな手法といえるでしょう．なお，FOR～NEXTに使う変数は，ここではN％(＝30)ですが，たとえばFOR I％＝1 TO 20と，N％より小さい数を使う時はエラーを起しません．その代り，たとえば FOR I％＝1 TO 32などとN％よりも大きい数を使うと，“Input past end”……もうデータがないのに読出し命令をしています，というエラーになります．

＊＊＊とりあえずデータを読出してみる

データがどれだけ収められているかが分らずにファイル操作をすることは，原則としてあり得ないことですが，さきほどのデータを，FOR～NEXTを使わずに画面に出してみましょう．それが下段のプログラムです．ディスケットに収められているN％は単独で読出し，さきほどの30個のデータ群を画面に出してみます．

これが行番号150～170です．WHILE NOT EOF (1) は（ファイルにデータがある間は）という意味です．EOF（エンドオブ・ファイ

配列変数をファイルに書く▶

```
100 '====== PRINT#,INPUT#  LESSON 2 ========
110 '                    ﾃﾞｰﾀｦ ｶｸ
120 DIM X(30):N%=30:CLS '･･･････N%=ﾃﾞｰﾀﾉｶｽﾞ
130   FOR I%=1 TO N%
140     READ X(I%)
150   NEXT I%
160 '
170 OPEN"O",#1,"SAMP.X30
180 PRINT #1,N%  '･･･ﾃﾞｰﾀﾉｶｽﾞｦ ﾀﾝﾄﾞｸﾃﾞ ｶｷｺﾑ
190 FOR I%=1 TO N%:PRINT #1,X(I%):NEXT I%
200 CLOSE
210 '
220 END
230 '
240 DATA 1,5,88,54,33,351,258,952,541,120
250 DATA 3,62,80,44,55,85,44,200,31,-520
260 DATA 23,12,805,414,255,-805,84,20,15,-10
```

上の逆操作で
ファイルからデータを読む▶

```
100 '===== PRINT#,INPUT#  LESSON 2-1 =====
110 '          ﾃﾞｰﾀｦ ﾖﾑ *1
120 WIDTH80,20:CLS
130 OPEN"I",#1,"SAMP.X30
140 INPUT #1,N%:DIM X(N%)'ﾃﾞｰﾀｦﾖﾐ ﾊｲﾚﾂｼﾃｲ
150 FOR I%=1 TO N%
155   INPUT #1,X(I%):PRINT X(I%),
156 NEXT I%
160 CLOSE
170 '
180 END
```

EOF関数を使う▶

```
100 '======= PRINT#,INPUT#  LESSON 2-2 =======
110 '             ﾃﾞｰﾀｦ ﾖﾑ *2
120 WIDTH 80,20:CLS
130 OPEN"I",#1,"SAMP.X30
140 INPUT #1,N%
150 WHILE NOT EOF(1) '･････ﾌｧｲﾙﾆ ﾃﾞｰﾀｶﾞｱﾙｱｲﾀﾞ-┐
160 . INPUT #1,X:PRINT X,                :'     |
170 WEND '･･･････････････････INPUT# ｦ ｼﾞｮｺｳ----┘
180 CLOSE
190 '
200 END
```

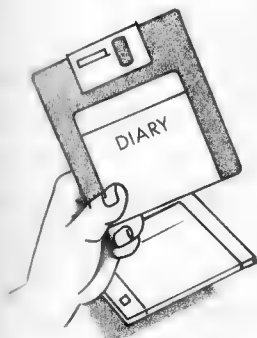
ル）は関数で，ファイルが終らない間は偽を返してきます．したがって，NOTを使うことで「ファイルが終らない間は真」になり，WENDではさんだ部分の命令を実行します．

※Input past endエラーが起きたら，いったんファイルを閉じなければいけない

データの読み，書き，訂正，消去

＊まず，簡単な例でファイル操作を

もっとも簡単なファイル操作を考えましょう．とりあえず「フロッピー日記」ということにして，年・月・日を入力し，それをファイル名にし，１回ごとにひとつの文字列を記録してゆきます．ファイルへの書きこみ，読出しについてはすでに知っているはずですから，ここでは説明しません．それぞれをサブルーチンにして，必要に応じて呼び出すようにしています．ファイル名は“NK”＋YMD＄になり，YMD＄は６ケタの数字で入力しますから，最終的なファイル名はたとえば84年12月１日なら“NK841201”といったものになり，この名前から逆に日付を見つけます．

```
100 '================= * *   FLOPPY DIARY   * * =======================
110 '
120 WIDTH 40,20:CLS
130   A$="******* M E N U ********":LOCATE 0,4:GOSUB 650
140   A$="1･･･････････････ﾆｯｷｦ ｶｸ":LOCATE 0,6:GOSUB 650
150   A$="2･･････････ﾆｯｷﾉ ﾃｲｾｲ":LOCATE 0,7:GOSUB 650
160   A$="3･･･････････････ﾆｯｷｦ ﾐﾙ":LOCATE 0,8:GOSUB 650
170   A$="4･･･････････････ﾆｯｷｦ ｹｽ":LOCATE 0,9:GOSUB 650
180   A$="5･･････････････ｼｭｳﾘｮｳ":LOCATE 0,10:GOSUB 650
190   A$="**************************":LOCATE 0,12:GOSUB 650
200  A$="ﾄﾞﾚﾆ ｼﾏｽｶ ﾊﾞﾝｺﾞｳﾃﾞ ﾄﾞｳｿﾞ ?":LOCATE 0,15:GOSUB 650
210    MENU$=INPUT$(1):MENU=VAL(MENU$):IF MENU<1 OR MENU>5 GOTO 210
220 ON MENU GOTO 240,320,390,460,500
230 '
240 CLS:PRINT"*********** ﾆｯｷﾉ ｶｷｺﾐ **************"
250 GOSUB 680 '･･････････････････････････････････････YMD INPUT
260 A$="ﾌﾞﾝｼｮｳｦ ﾄﾞｳｿﾞ(MAX 255 ﾓｼﾞ)":LOCATE 0,0:GOSUB 650
270   INPUT NIKKI$
280 GOSUB 550 '･･････････････････････････････････････ﾌｧｲﾙ ｻｸｾｲ
290 GOTO 120 '･･･････････････････････････････････････ﾒﾆｭｰﾍ ﾓﾄﾞﾙ
300 '
310 '
320 CLS:PRINT"*********** ﾆｯｷﾉ ﾃｲｾｲ **************"
330 GOSUB 680 '･･････････････････････････････････････YMD INPUT
340 GOSUB 600 '･･････････････････････････････････････ﾌｧｲﾙｶﾗﾉ ﾖﾐﾀﾞｼ
350 PRINT:PRINT NIKKI$ '･････････････････････････････ﾓﾄﾉﾌﾞﾝｼｮｳ
360 PRINT:PRINT"ﾃｲｾｲﾉ ﾌﾞﾝｼｮｳｦ ﾄﾞｳｿﾞ":INPUT NIKKI$ '･･･ﾃｲｾｲﾌﾞﾝ
370 KILL"0:"+YMD$:GOSUB 550 '････････････････････････ｶｷｺﾐ
380 GOTO 120'････････････････････････････････････････ﾒﾆｭｰﾍﾓﾄﾞﾙ
390 CLS:PRINT"*********** ﾆｯｷｦ ﾐﾙ **************"
```

マイクロフロッピーに日記をつける

内容の訂正は，まず元のデータを読出して画面に出し，訂正したい文字を求めたら，元のファイルを消してから，新しい内容を書きこみます．内容を消すときはKILL文を使いますが，プログラムの中でファイルを消すときは“Are you sure？”というメッセージは出ずに，プログラムにしたがって消してしまいます．

なお，メニューの1～4までのルーチンは，それぞれが終了するとメニューへもどるようになっています．メニューの5を指定するとファイルをCLOSEして，プログラムの実行を終了します．全体で70行ほどのリストですが，それぞれのルーチンは短かいのでプログラムの流れはすぐに分ると思います．画面に出す文字列を左右の中央に出すために，行番号650で40桁から文字列の長さ分を引き，それを半分にした空白の文字列を利用しています．

```
400 GOSUB 680 '........................................YMD INPUT
410 GOSUB 600 '........................................ﾌｧｲﾙｶﾗﾉ ﾖﾐﾀﾞｼ
420 PRINT:PRINT NIKKI$
430 PRINT:PRINT"OK?  ﾒﾆｭｰﾍ ﾓﾄﾞﾙﾅﾗ ANY KEY PUSH"
440 IF INKEY$<>"" THEN 120 ELSE 440
450 '
460 CLS:PRINT"*********** ﾆｯｷｦ ｹｽ **************"
470 GOSUB 680 '........................................YMD INPUT
480 KILL"0:"+YMD$:GOTO 120 '..........................ﾒﾆｭｰﾍ ﾓﾄﾞﾙ
490 '--------------------- THE END ---------------------------------
500 CLS:A$="T H E  E N D":LOCATE 0,10:GOSUB 650
510 CLOSE:END
520 '
530 '************************ ｻﾌﾞﾙｰﾁﾝ ******************************
540 '---------------------------- ｶｷｺﾐ ----------------------------
550 OPEN"O",#1,"0:"+YMD$
560 PRINT #1,NIKKI$
570 CLOSE:RETURN
580 '
590 '---------------------------- ﾖﾐﾀﾞｼ ---------------------------
600 OPEN "I",#1,"0:"+YMD$
610  INPUT#1,NIKKI$
620 CLOSE:RETURN
630 '
640 '--------- ﾓｼﾞﾉ ｼｮﾘ
650 PRINT SPACE$((40-LEN(A$))/2);A$:RETURN  '............ﾓｼﾞﾉ ｾﾝﾀｰｿﾞﾛｴ
660 '
670 '-------- YMD$ INPUT
680 CLS:INPUT"ﾅﾝ年ﾅﾝ月ﾅﾝ日 ﾃﾞｽｶ(ﾚｲ 840909)";YMD$:IF LEN(YMD$)<>6 THEN 680
690 YMD$="NK"+YMD$:RETURN
```

ランダムファイルの基本

*セクタ単位で記録

ランダムファイル（random access file）は，ひとつのデータ群をまとまりのあるものとして扱い，それぞれにレコード番号（データを何番目に記録したかの目印）をつけるものです．データを取出したり，訂正する時などはこのレコード番号を使って直接操作できるというのが最大の利点になっています．その反面，1データの記録用に1セクタ（sector：ディスケットの区画の単位）を使いますから，1文字分を記録しても，255文字分を記録しても一定の区画を使ってしまいます．また数値データは記録できないことや，1データ群に複数のデータがある時（例：住所と電話番号と氏名など）などは，このセクタをどう使うかをあらかじめ決めておかなければならないなど，いろいろな手続きをしなければいけません．

※工夫をすれば，1セクタに複数のデータを詰めこむことができる

また，コンピュータのデータはバッファ（buffer：緩衝装置．電子回路同士や，回路とメカニズム部などが互いに干渉しないための回路や装置というのが原義）を通じてディスクドライブとのやりとりをするのですが，ランダムファイルではデータのやりとりをGET・PUTという命令で実行します．

**PUTでデータが書きこまれる

まずランダムファイルの基本をしっかりと頭に入れましょう．

OPEN "R", #1, "レンシュウ"

とタイプして，リターンキーを押してください．これだけでファイルが作られ，ディスケットには“レンシュウ”というファイルが作成されてます．このデータが書かれていないファイルは

KILL "レンシュウ"

で消すことができます．ランダムファイルの作成・読出しのもっとも簡単な例をサンプルに示します．データの書きこみを命令するのは

PUT 〈ファイル番号〉,〈レコード番号〉

で，これが実行されるとバッファにあるデータがディスケットへと転送されて記録されます．このプログラムではN=N+1と，レコード番号は1ずつ増えており，記録も1，2，3，……という番号で書きこまれてゆきます．なお，記録用のXR$は200バイト分を設定し，左づめ指定なので，画面に出力した時も左づめになります．

仕切りのある箱からりんごやみかん，ももを取出してディスケットへ移す

***GETでデータを取出す

サンプルの上のプログラムを実行し，“TEST” というファイルを作ったら，このプログラムを適当な名前，たとえば “PUT1” などとして，SAVEしてください．つぎに下のプログラムを作成し，これを実行すると，“TEST” のファイル名のもとに何個分のデータがあるのかをまず知らせてくれます．そのデータ群の中の何番目が見たいのかを数字で入力するとバッファには指定のデータが送られます．それが行番号160のGET文です．

GET 〈ファイル番号〉,〈レコード番号〉

を実行すると，指定のものが取出されます．

なお，書きこみのプログラムではレコード番号を1きざみで指定しましたが，たとえばサンプルの上のプログラムで

140 N=N+10

と，10きざみにすると，見かけ上記録数が増えてしまいます．書きこみの時，3つのデータだけを記録したときもN=10, 20, 30で，最後は30ですから，下の読出しプログラムでも，LOF（レンクスオブファイル）は「30」という数値を返してくるわけです．ランダムファイルもそのフォーマット（format：形式）を守れば，難しくありません．

```
100 '---------------------- ﾗﾝﾀﾞﾑ ﾌｧｲﾙ 1/ｶｷｺﾐ ----------------------------------
110 N=0:WIDTH 80,20:CLS
120 OPEN"R",#1,"0:TEST" '························TEST ﾄｲｳ ﾌｧｲﾙﾒｲﾃﾞﾗﾝﾀﾞﾑﾌｧｲﾙ ｼﾃｲ
130 FIELD #1,200AS XR$ '························200ﾊﾞｲﾄﾉXR$ｦ ｷﾛｸﾖｳﾆ ﾜﾘｱﾃﾙ
140 N=N+1
150 INPUT"ﾃｽﾄﾖｳﾉ ｺﾄﾊﾞｦ ﾄﾞｳｿﾞ";X$:LSET XR$=X$ '···ﾆｭｳﾘｮｸｦ ﾋﾀﾞﾘﾂﾞﾒﾆ ｼﾃｲ
160 INPUT"ﾂﾂﾞｸﾙ·····1 ｵﾜﾘ·····2"; Y
170 IF Y=1 THEN PUT#1,N:GOTO 140 '················1ﾚｺｰﾄﾞﾌﾞﾝｦ ｶｷｺﾑ
180 IF Y=2 THEN PUT#1,N:CLOSE:END '···············1ﾚｺｰﾄﾞﾌﾞﾝｦ ｶｷｺﾝﾃﾞ ｵﾜﾙ
190 GOTO 160


100 '----------------------- ﾗﾝﾀﾞﾑ ﾌｧｲﾙ 1/ﾖﾐｺﾐ ----------------------------------
110 WIDTH 80,20:CLS
120 OPEN"R",#1,"0:TEST" '························TEST ﾄｲｳﾌｧｲﾙﾒｲﾉ ﾗﾝﾀﾞﾑﾌｧｲﾙｦ ｼﾃｲ
130 FIELD #1,200AS XR$ '·························200ﾊﾞｲﾄﾉ XR$ｦ ﾊﾞｯﾌｧｶﾞ ｳｹﾂｹﾙ
140 NUM=LOF(1):PRINT"ｷﾛｸﾃﾞｰﾀ";NUM '···············ｷﾛｸｻﾚﾃｲﾙ ﾚｺｰﾄﾞﾉ ｶｽﾞｦ ﾓﾄﾒﾙ
150 INPUT"ﾅﾝﾊﾞﾝｦ ﾐﾏｽｶ";N:IF N<1 OR N>NUM THEN 150
160 GET#1,N:PRINT XR$ '··························ﾚｺｰﾄﾞﾉ ﾊﾞﾝｺﾞｳｦ ｼﾃｲｼ ﾃﾞｰﾀｦﾋｮｳｼﾞ
170 INPUT"ﾂﾂﾞｸﾙ·····1 ｵﾜﾘ·····2";Y
180 IF Y=1 THEN 150
190 IF Y=2 THEN CLOSE:END
200 GOTO 170
```

FIELD文とL／RSET文

＊区画を設定するFIELD文

ランダムファイル用のバッファはフロッピーの1セクタと1対1の対応をしています．256バイト分をどういう形で区画するのか，それぞれの1区画にどういう形でデータを収めておくかを，あらかじめセットしておかなければいけません．それがフィールド（field）文です．フィールドは野原，分野といった意味ですが，ここでは区画分けとでも解釈しておきましょう．

FIELD 〔ファイル番号〕,〔設定バイト数　AS　変数名〕

という書きかたをすると，FM-77はそれに従って，変数名をそれぞれの区画にセットし，PUT文が実行されるとディスケットへと書込みます．

もうひとつ，LSET／RSET文で文字列を左づめにするか右づめにするかを指定します．この2つの操作をしてからPUTすれば，データがフロッピーに書込まれます．

＊＊LSETとRSET

サンプルプログラムを見てください．

ここでは3つの文字列を20バイトずつ，3つの区画にしていますが2番目はRSET，それ以外はLSETにしています（行番号140～150）．RSETとLSETの実際を見てみましょう．実行例の「3つを出力」を見ると，LSETは画面の左から出力され，RSETは“変数の先頭から”20文字分で切捨てられて，右づめ出力になっていることが分ります．

これをさらに明確にしたのが「3つをひとかたまりでGET」したもので，ここではGETするためのフィールドを60バイト分に設定して，ひとつの変数名で，バッファからデータを取出しています．結果は実行例を見ても分るように，フロッピーに書かれたデータが左づめ・右づめ・左づめの形で出力されました(行番号280～300)．

“コンニチハ”というデータは

文字列の“コンニチハ”＋15字分の空白の文字列

という形でフロッピーに入っていて，“コンニチハ”の右側は単なる余白になっているわけでなく，文字列になっているのです．これを確認するにはRSETのデータを，PRINT　USING文で出力するとよい

でしょう．たとえばFIELD文で5バイトに設定し，RSET操作をしたデータ（変数には“ミホン”が代入されているとして）は

□□ミホン

という形でフロッピーに書かれています．PRINT　USING文を使っても，やはり「　　ミホン」という形で出力され，ふつうの変数のような「ミホン　　」にはなりません．なお，サンプルプログラムではレコード番号を指定していませんが実行のたびに1ずつ増えます．

下のプログラムの実行例▶

```
ﾂﾂﾞﾘ  1? ｺﾝﾆﾁﾊ
ﾂﾂﾞﾘ  2? ﾜﾀｼﾊ 8ﾋﾞｯﾄ ﾊﾟｰｿﾅﾙｺﾝﾋﾟｭｰﾀﾉ
ﾂﾂﾞﾘ  3? FM-77 ﾃﾞｽ｡ﾖﾛｼｸ
3ﾂﾉ ﾂﾂﾞﾘｦ ﾌｧｲﾙﾆ ｼﾏｲﾏｼﾀ｡ﾂｷﾞﾊ ﾌｧｲﾙｶﾗ ﾄﾘﾀﾞｼﾏｽ
         PUSH ANY KEY


3ﾂｦ  ｼｭﾂﾘｮｸ
ｺﾝﾆﾁﾊ
ﾜﾀｼﾊ 8ﾋﾞｯﾄ ﾊﾟｰｿﾅﾙｺﾝﾋ
FM-77 ﾃﾞｽ｡ﾖﾛｼｸ
3ﾂｦ ﾋﾄｶﾀﾏﾘﾃﾞ GET
ｺﾝﾆﾁﾊ                    ﾜﾀｼﾊ 8ﾋﾞｯﾄ ﾊﾟｰｿﾅﾙｺﾝﾋFM-77 ﾃﾞｽ｡ﾖﾛｼｸ
```

```
100 '**************** RANDOMFILE  FIELD & L・R SET ***************************
110 WIDTH 80,20:OPEN "R",#1,"0:TEST2":'・・・・・・・・・・・・・・・・・・・・・・・ﾗﾝﾀﾞﾑﾌｧｲﾙ ｦ OPEN
120 FOR I=1 TO 3:PRINT USING"ﾂﾂﾞﾘ ##";I;:INPUT IN$(I):NEXT:'・・・・ﾆｭｳﾘｮｸ
130 'FIELD ﾄ L/R SET
140     FIELD #1,20AS RIN1$,20AS RIN2$,20AS RIN3$
150     LSET RIN1$=IN$(1):RSET RIN2$=IN$(2):LSET RIN3$=IN$(3)
160 PUT #1:CLOSE:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾌﾛｯﾋﾟｰﾆ ｶｷｺﾑ
170 'ｾﾂﾒｲ
180         PRINT"3ﾂﾉ ﾂﾂﾞﾘｦ ﾌｧｲﾙﾆ ｼﾏｲﾏｼﾀ｡ﾂｷﾞﾊ ﾌｧｲﾙｶﾗ ﾄﾘﾀﾞｼﾏｽ"
190         PRINT"          PUSH ANY KEY"
200         WHILE INKEY$="":WEND:PRINT:PRINT
210 'ｶｷｺﾐﾄ ｵﾅｼﾞ FIELD
220      OPEN "R",#1,"0:TEST2"
230 N=LOF(1):'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ｻｲｼﾝﾉ ﾃﾞｰﾀｦ ﾄﾘﾀﾞｽ
240      FIELD #1,20AS RIN1$,20AS RIN2$,20AS RIN3$
250 GET #1,N:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾌﾛｯﾋﾟｰｶﾗ  ﾖﾑ
260 'ｶｷｺﾐﾄ ｺﾄﾅﾙ FIELD
270            PRINT"3ﾂｦ  ｼｭﾂﾘｮｸ":PRINT RIN1$:PRINT RIN2$:PRINT RIN3$
280      FIELD #1,60AS RIN$
290 GET #1,N:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾌﾛｯﾋﾟｰｶﾗ  ﾖﾑ
300            PRINT"3ﾂｦ ﾋﾄｶﾀﾏﾘﾃﾞ GET":PRINT RIN$
310 '
320 CLOSE
```

1セクタに多くのデータを詰める

＊1セクタは1レコードに対応しているが……

ランダムファイルは，1セクタに1レコード番号をつけるのが原則ですが，ひとつのデータが40バイト程度の時，ふつうの使いかたではムダが出てしまいます．フロッピーを効率よく使うために，データを詰めることを考えましょう．フロッピーに記録する真のレコード番号(TN)と，使い手が指示するレコード番号(PN)は，たとえば1セクタに6つのデータを詰める場合は

TN＝1　PN＝1～6　　　TN＝2　PN＝7～12
TN＝3　PN＝13～18　　…………

というようになります．PNからTNを求める計算式は

TN＝(PN－1) ￥DN＋1 (DNは1セクタに詰めるデータ数)

で求めることができます．ランダムバッファからフロッピーへとデータを書込むPUT文にはこのTNを使います．

＊＊レコード番号とFIELD文の関係

もうひとつ，FIELD文のほうも考えなければいけません．レコード番号とバッファとの関係は，たとえばDN＝6のときは

レコード番号＝1，7，13，19，…：左はしにFIELDをセット
レコード番号＝2，8，14，20，…：左から2番目に　〃
レコード番号＝3，9，15，21，…：左から3番目に　〃
…………　　…………　　…………

レコード番号＝6，12，18，24…：右はしにFIELDをセットというようにします．1セクタにおけるデータの位置をRNとすると，計算式は

RN＝(PN－1) MOD DN ＋1

になります．サンプルプログラムではデータは配列変数に入力し，この変数をバッファのSET文と対応させています．たとえばDN＝6のとき，レコード番号15ならRN＝3になり，DAT＄(3)はバッファの左から3番目にセットされるわけです．

なお，6個分のデータがバッファに入ったらそこでPUTして書込みます．それと同時に配列変数を消して，前のデータがメモリに残らないようにしています．この操作をしないと，入力を終えたとき，前のデータが配列に残ったままPUTされるからです．

```
100 '==================== RANDOM FILE ** BLOCK PUT ** =========================
110 '                              ﾃﾞｰﾀﾉ ｼﾝｷ ｻｸｾｲ
120 CLEAR 800:DEFINT A-Z:DN=6:PN=1:BYTE=40:DIM DAT$(DN)
130 REM DN=1ｾｸﾀﾅｲﾉ ﾃﾞｰﾀﾉ ｶｽﾞ  BYTE=1ﾃﾞｰﾀﾍﾉ ﾜﾘｱﾃ ﾊﾞｲﾄｽｳ
140 OPEN "R",#1,"0:BLOCK":FIELD #1,BYTE AS D1$,BYTE AS D2$,BYTE AS D3$,BYTE AS D
4$,BYTE AS D5$,BYTE AS D6$
150 '
160 TN=(PN-1)¥DN +1:RN=(PN-1)MOD DN +1:'·····ｼﾝﾉ ﾚｺｰﾄﾞﾊﾞﾝｺﾞｳﾄ ｾｸﾀ ﾅｲﾉ ﾊﾞﾝｺﾞｳ
170 PRINT USING "NO.=#####";PN:INPUT"DATAｦ ﾄﾞｳｿﾞ";DAT$(RN)
180 LSET D1$=DAT$(1):LSET D2$=DAT$(2):LSET D3$=DAT$(3):LSET D4$=DAT$(4):LSET D5$
=DAT$(5):LSET D6$=DAT$(6)
190 IF RN MOD DN=0 THEN PUT #1,TN:ERASE DAT$:DIM DAT$(DN)
200 PN=PN+1:'·································ﾚｺｰﾄﾞﾊﾞﾝｺﾞｳｦ ﾋﾄﾂ ﾌﾔｽ
210 PRINT"ﾂﾂﾞｹﾙ･･･>ANY KEY  ｵﾜﾘ･･･>E"
220 YN$=INPUT$(1)
230 IF YN$="E" OR YN$="e" THEN PUT #1,TN:CLOSE:END
240 GOTO 160


100 '==================== RANDOM FILE ** BLOCK GET ** =====================
110 '                                    ﾃﾞｰﾀｦ ﾐﾙ
120 DEFINT A-Z:DN=6:BYTE=40:OPEN "R",#1,"0:BLOCK"
130 REM DN=1ｾｸﾀﾅｲﾉ ﾃﾞｰﾀﾉ ｶｽﾞ  BYTE=1ﾃﾞｰﾀﾍﾉ ﾜﾘｱﾃ ﾊﾞｲﾄｽｳ
140 FIELD #1,DN*BYTE AS D$:'·····················240ﾊﾞｲﾄ ﾌﾞﾝﾉ FIELD ｾｯﾃｲ
150 L=LOF(1):'·······························ｻｲﾀﾞｲ ﾚｺｰﾄﾞ ﾊﾞﾝｺﾞｳｦ ﾓﾄﾒﾙ
160 INPUT"ﾚｺｰﾄﾞ ﾊﾞﾝｺﾞｳ";PN:IF PN=0 THEN BEEP:GOTO 160
170 TN=(PN-1)¥DN+1:RN=(PN-1) MOD DN +1:'········ｼﾝﾉ ﾚｺｰﾄﾞﾊﾞﾝｺﾞｳﾄ ｾｸﾀ ﾅｲﾉ ﾊﾞﾝｺﾞｳ
180 IF  TN>L THEN BEEP:PRINT"ﾃﾞｰﾀﾊ ｱﾘﾏｾﾝ":GOTO 160
190 GET #1,TN:'·································ﾖﾐｺﾐ
200 'ｼﾃｲﾉ ﾃﾞｰﾀｦ ｻｶﾞｽ
210 IF RN=1 THEN PRINT LEFT$(D$,BYTE) ELSE PRINT MID$(D$,(RN-1)*BYTE+1,BYTE)
220 PRINT"ﾂﾂﾞｹﾙ･･･>ANY KEY  ｵﾜﾘ･･･>E"
230 YN$=INPUT$(1):IF YN$="E" OR YN$="e" THEN END ELSE 160
```

***GET文について

こうして，レコード番号の1から連続して記録されたデータを読むことを考えましょう．サンプルでは40バイトずつ，6個分のデータが入っているので，FIELD文は240バイトにしています（配列変数を使う必要はありません）．探したいレコード番号を入力すると，やはり真のレコード番号と，セクタ内の位置（ランダムバッファの位置と1対1に対応）を計算し，行番号210のように文字列関数を使って目的のものを取出しています．

これらのサンプルを中心にデータの追加／訂正プログラムを作れば本格的に使いこなせるものになります．PUT，GETはセクタ単位の操作であることに注意すれば，あとは難しくありません．

わが家の蔵書を一覧表に

＊蔵書の内容をファイル化する

私たちの身のまわりにはいろいろなコレクションがあります．本やレコード・カセット，ビデオソフト，衣類や女性のアクセサリーなどもコレクションと呼べないこともありません．これらをファイルにしまっておけば，どこに何があるのか，いつ購入したのかなどのさまざまなデータを見ることができるので，いちいち探しものをする必要もなく，また品物によっては耐用年数がいつなのかを知ることができるといったことも可能です．

ランダムファイルを使った例として，ここでは本を取上げてみました．タイトル，著者名，出版社名，内容などを記録して，一覧表として見ることもできますし，プリントアウトも可能なものです．

＊＊5つのメニューを用意

プログラムの基本設計としては

1 データの作成
2 データの追加
3 データの訂正（内容の画面出力を含む）
4 データのプリントアウト

の4項目とします．データをさらに有効に使うためには

○データの検索（例：同一出版社の本ごとに出力する）
○データのソート（例：出版年度別に出力する）

などのプログラムを考えるとよいでしょう．これらについては，本書にその基本テクニックがありますから，それをもとにして発展させてみましょう．

これらをもとにして，まずメニュー画面を作成します．どの作業を選ぶのかをINKEY＄で求め，ここではそれをいちど数値化してから（行番号260），それぞれの指定に従ったサブルーチンを呼び，ひと仕事終えたらまたメニュー画面に戻しています（行番号270～280）．なお，サブルーチンのRETURN文は戻り先の指示をしない書き方と，指示をする書き方があり，たとえば「新規作成」において行番号330を「RETURN　150」というように書くこともできます．メニューで選んだサブルーチンのすべてを行番号150に戻すと，行番号280は不必要になります．

※ファイル操作を伴なうプログラムの実行中に「BREAK」をかけたら，かならずCLOSEをしてからデバックをしたり，再開をする

```
                        ****** ﾜｶﾞﾔﾉ ｿﾞｳｼｮ *****

------------------------------------------------------------------------------
ﾊﾞﾝｺﾞｳ= 2      1-ﾎﾝﾉﾀｲﾄﾙ=ﾊｼﾚ ﾄﾏﾎｰｸ                  2-ﾁｮｼｬ=ﾔｽｵｶ ｼｮｳﾀﾛｳ
3-ｼｭｯﾊﾟﾝｼｬ=ｺｳﾀﾞﾝｼｬ                 4-ﾍﾟｰｼﾞｽｳ=229      5-ﾊｯｺｳﾋﾞ=S48.9.24
6-ﾃｲｶ=780円                         7-ﾎﾝﾉ ﾌﾞﾝﾙｲ=ｼﾞｭﾝﾌﾞﾝｶﾞｸ
8-ﾒﾓ
ﾀﾝﾍﾟﾝｼｭｳ

------------------------------------------------------------------------------
ﾊﾞﾝｺﾞｳ= 3      1-ﾎﾝﾉﾀｲﾄﾙ=ｾｲﾎｳﾉ ｵﾄ                    2-ﾁｮｼｬ=ｺﾞﾐ ﾔｽｽｹ
3-ｼｭｯﾊﾟﾝｼｬ=ｼﾝﾁｮｳｼｬ                 4-ﾍﾟｰｼﾞｽｳ=280      5-ﾊｯｺｳﾋﾞ=1969.7.30
6-ﾃｲｶ=700円                         7-ﾎﾝﾉ ﾌﾞﾝﾙｲ=ｽﾞｲﾋﾂ
8-ﾒﾓ
ｹﾝｺﾞｳ ｻｯｶﾉ ｵﾝｶﾞｸ･ｵｰﾃﾞｨｵ ｽﾞｲﾋﾂｼｭｳ

------------------------------------------------------------------------------
ﾊﾞﾝｺﾞｳ= 4      1-ﾎﾝﾉﾀｲﾄﾙ=ｳﾀﾉ ｼｮｳﾜｼ                   2-ﾁｮｼｬ=ｶﾀ ｺｳｼﾞ
3-ｼｭｯﾊﾟﾝｼｬ=ｼﾞｼﾞ ﾂｳｼﾝｼｬ              4-ﾍﾟｰｼﾞｽｳ=284      5-ﾊｯｺｳﾋﾞ=S50.9.1
6-ﾃｲｶ=980円                         7-ﾎﾝﾉ ﾌﾞﾝﾙｲ=ﾌｳｿﾞｸ
8-ﾒﾓ
ｶﾖｳｷｮｸﾆﾐﾙ ｼｮｳﾜﾉ ｾｿｳ｡<<ﾊﾌﾞﾉ ﾐﾅﾄ>> ｶﾗ <<ﾊﾊﾆｻｻｹﾞﾙ ﾊﾞﾗｰﾄﾞ>> ﾏﾃﾞ
```

▲データのプリントアウト例（番号2〜4までを指定したとき）

```
100 '======================== ﾗﾝﾀﾞﾑ ﾌｧｲﾙ    ﾜｶﾞﾔﾉ ｿﾞｳｼｮ ======================
110 '=                          ** ﾌｧｲﾙﾒｲ=BOOK **                              =
120 '==========================================================================
130      DEFINT A-Z:FOR I=1 TO 8:READ TITLE$(I):NEXT '･････････････ﾀｲﾄﾙﾉ ﾖﾐｺﾐ
140 '---------------------- ﾒﾆｭｰ -----------------------------
150 WIDTH 40,20:CLS:LOCATE 0,2
160   PRINT"      **********************************":PRINT
170   PRINT"             M  E  N  U":PRINT
180   PRINT"          1･･････ﾃﾞｰﾀﾉ ｼﾝｷ ｻｸｾｲ":PRINT
190   PRINT"          2･･････ﾃﾞｰﾀﾉ ﾂｲｶ":PRINT
200   PRINT"          3･･････ﾃﾞｰﾀﾉ ﾃｲｾｲ":PRINT
210   PRINT"          4･･････ﾃﾞｰﾀﾉ PRINT OUT":PRINT
220   PRINT"          5･･････ｻｷﾞｮｳ ｵﾜﾘ":PRINT
230   PRINT"      **********************************":PRINT
240   PRINT"    ﾄﾞﾉ MENU ｦ ｴﾗﾋﾞﾏｽｶ"
250   MENU$=INKEY$:IF MENU$="" THEN 250
260   MENU=VAL(MENU$):IF MENU<1 OR MENU>5 THEN 260
270 ON MENU GOSUB 310,570,620,1160,1520
280 GOTO 150 '･･････････････････････････････ﾆｭｳﾘｮｸ ﾌﾃｷﾄｳﾅﾗ MENUﾍ ﾓﾄﾞﾙ
```

＊＊＊データの新規作成

データの新規作成では

1 ファイル名を指定すること

2 項目ごとに何文字分を設定するのか，左づめにするのか右づめにするのかなどのフォーマットを決めること

3 ひとつのデータ群の書きこみを終えたら，レコード番号に注意して，書きこみを続けるか，終えるかを決めること

などに注意します．また，INPUT文はとくに重要で，通常のINPUT文にするのか，LINE INPUT文にするのかなどを考えておきます．本のタイトルその他にダブルコーテーション（"）やコロン（：）などがつけられていることも考えて，ここではすべての入力文をLINE INPUT型にしています．

＊＊＊＊データの追加と訂正

つぎのデータの追加は，いままでにも出てきたように，レコード数をまず求め，そこに＋1をしてゆくだけの作業のほかは，データの新規作成とまったく同じですから，「データの作成」部分は共有プログラムが使えます．したがってこのプログラムでも作成部分はサブルーチン化（行番号360～540）しています．

データの訂正はすこし複雑になります．訂正プログラムは

○書きこまれたデータがどのレコード番号なのかを使い手が知っているかどうか

○ひとつのデータ群の訂正の時，そのデータ群の一部分を訂正できるようにするかどうか

○1回の作業で，同時に複数の訂正ができるようにするのか，1回の作業で1部分だけの訂正にするのか

などについて，あらかじめどう対応するのかを考えておかなければいけません．ここでは

1 レコード番号が指定できる場合とできない場合に分けた．訂正したいレコード番号が分らない時は，作成ずみのデータを始めから呼出して，発見できたら訂正プログラムへ戻るようにしている（行番号640・行番号700・行番号1010～1130）．

2 データを部分的に訂正できるようにし，1回の作業で1項目だけを訂正する

という形式をとっています．なおファイルを壊さないためにも，なるべく多くの箇所に「取消し」ルーチンを入れておきます．

```
290 '
300 '========================== 1 ｼﾝｷ ｻｸｾｲ ==============================
310 N=1 '·································ｻｲｼｮﾉ ﾃﾞｰﾀｽｳ=1ｶﾗ ﾊｼﾞﾒﾙ
320   GOSUB 360 'ﾃﾞｰﾀ ｻｸｾｲ SUB
330 RETURN
340 '
350 '*********** ﾃﾞｰﾀ ｻｸｾｲ SUB
360 WIDTH 80,20:CLS
370   GOSUB 1600 '········································OPEN SUB
380                                   NUM$=STR$(N):LSET RNUM$=NUM$
390   PRINT"ｷﾛｸ ﾊﾞﾝｺﾞｳ=";N
400   PRINT TITLE$(1);"? ";:LINE INPUT;D1$:LSET RB$=D1$
410   PRINT TITLE$(2);"? ";:LINE INPUT;D2$:LSET RN$=D2$
420   PRINT TITLE$(3);"? ";:LINE INPUT;D3$:LSET RS$=D3$
430   PRINT TITLE$(4);"? ";:LINE INPUT;D4$:LSET RPG$=D4$
440   PRINT TITLE$(5);"? ";:LINE INPUT;D5$:LSET RH$=D5$
450   PRINT TITLE$(6);"? ";:LINE INPUT;D6$:LSET RPR$=D6$
460   PRINT TITLE$(7);"? ";:LINE INPUT;D7$:LSET RCD$=D7$
470   PRINT TITLE$(8);"? ";:LINE INPUT;D8$:IF D8$="" THEN D8$="ﾄｸﾆﾅｼ"
480                                         LSET RMEM$=D8$
490   PRINT"OK? 1････ﾂｷﾞﾉｶｷｺﾐ  2････ｼｭｳﾘｮｳ"
500   YN$=INKEY$:IF YN$="" THEN 500
510   YN=VAL(YN$):IF YN<1 OR YN>2 THEN 500
520   IF YN=1 THEN PUT#1,N:N=N+1:GOTO 380 '·················ﾂﾂﾞｹﾙﾅﾗ Nｦ ﾋﾄﾂ ﾌﾔｽ
530   PUT#1,N:CLOSE '·································ｵﾜﾘﾅﾗ Nﾊ ﾌｴﾅｲ
540 RETURN
550 '
560 '========================== 2  ﾃﾞｰﾀ ﾂｲｶ  ==============================
570 OPEN"R",#1,"0:BOOK":N=LOF(1):N=N+1:CLOSE
580   GOSUB 360 '······································ﾃﾞｰﾀ ｻｸｾｲ SUB
590 RETURN
600 '
610 '========================== 3 ﾃﾞｰﾀ ﾃｲｾｲ ===============================
620 WIDTH 80,20:CLS
630   PRINT"ｺﾚｶﾗ ｶｷｺﾐ ﾃﾞｰﾀﾉ ﾃｲｾｲｦ ｼﾏｽ｡ｷﾛｸﾊﾞﾝｺﾞｳｦ ｼﾃｲ ﾃﾞｷﾏｽｶ?"
640   PRINT"1････ﾃﾞｷﾙ  2････ﾃﾞｷﾅｲ    3････ﾄﾘｹｼ･ｼｭｳﾘｮｳ"
650   TEI$=INKEY$:IF TEI$="" THEN 650
660   TEI=VAL(TEI$):IF TEI<1 OR TEI>3 THEN 650
670   IF TEI=3 THEN CLOSE:RETURN
680   GOSUB 1600 '·······································OPEN SUB
690   MAXNUM=LOF(1) '·······························ｻｲﾀﾞｲ ｷﾛｸｽｳｦ ﾓﾄﾒﾙ
700   IF TEI=2 THEN 1010 '·······················ﾊﾞﾝｺﾞｳ ﾌﾒｲﾅﾉﾃﾞ ﾃﾞｰﾀｦﾐﾙ ﾌﾟﾛｸﾞﾗﾑﾍ
710 '
720 '********** ﾊﾞﾝｺﾞｳｼﾃｲﾉ ﾃｲｾｲ
730   PRINT:INPUT"ｷﾛｸﾊﾞﾝｺﾞｳｦ ﾄﾞｳｿﾞ";N:IF N<1 OR N>MAXNUM THEN PRINT"ﾊﾞﾝｺﾞｳ ﾏﾁｶﾞｲ
!!! ｼﾃｲｦ ﾀﾀﾞｼｸ":GOTO 730
740   GET#1,N:PRINT
750   GOSUB 940 '·····························ﾃﾞｰﾀﾋｮｳｼﾞ SUBﾍ
```

この訂正ルーチンはまた，記録ずみのデータを画面で見てゆくだけの作業にも使えます．具体的には行番号1040で，データ群を3組出力(IF I MOD 3＝0……という文．これによりデータはIが3，6，9……の時に表示を一時停止する）したら，スペースバーなどを押すと，また次のデータ群が表示されるわけです．こうしてすべてのデータを表示すると行番号620の訂正プログラムの始めに戻りますから，ここで「3」の「取消し・終了」を選べばデータは訂正されることなく，またメニューへと戻るわけです．

部分の訂正はすべてサブルーチン形式にして，訂正したいものを番号で指定（行番号760〜780）したら，指定のサブルーチンを呼び出し（行番号790）て訂正する方法です．

*****プリントアウト

プリントアウトのプログラムはそれほどむずかしくはありません．データのすべてを印字するのか，部分的に印字するのかだけが入力ルーチンになり，あとは指定に従うだけです．なお，ここではすべてのデータのプリントアウトと，部分のプリントアウトを分けた書きかたにしていますが，同一のプログラムでも可能です．たとえばメニューによってPOUTという変数は1か2のどちらかになり，1のときに「すべての蔵書のプリントアウト」，2のときは「部分の蔵書のプリントアウト」としましょう．1のときは

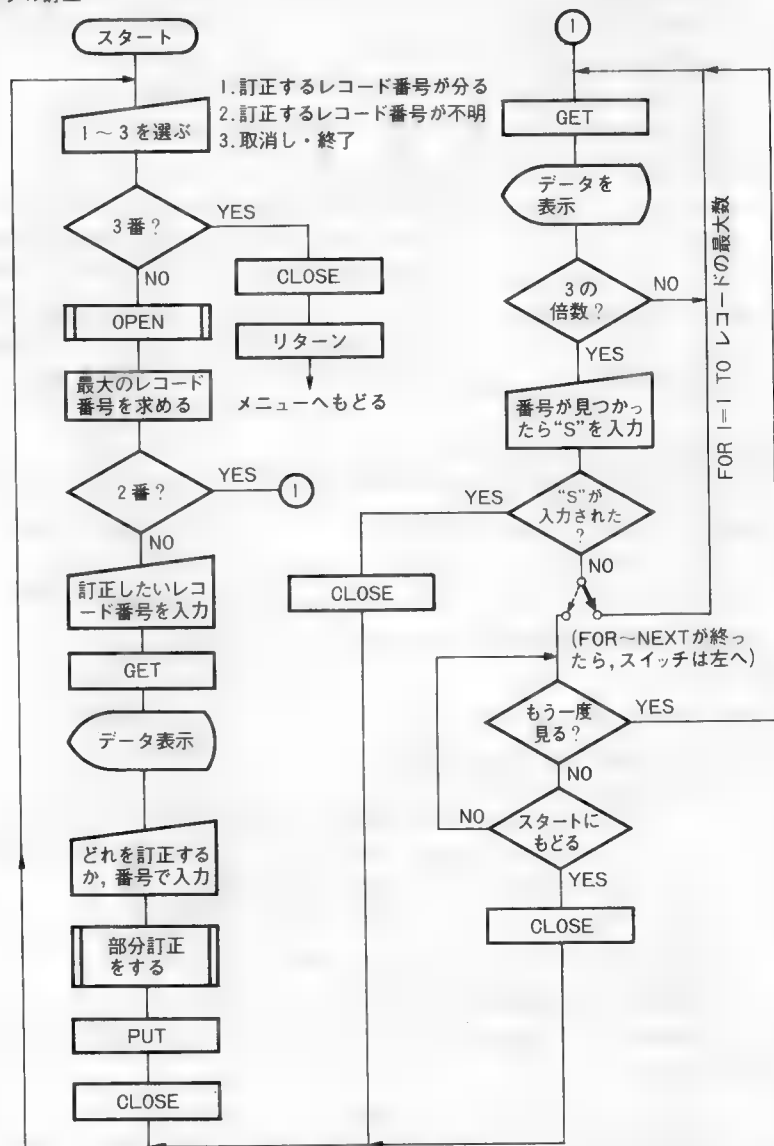
```
IF POUT=1 THEN STAR=1:STP=MAXNUM
```

という文を書いておくことで，こののちにプリントアウトのサブルーチンを呼び

```
FOR I=STAR TO STP :GET #1, I
  GOSUB 「プリントアウト」
NEXT I
```

とすればよいのです．このプログラムでは分りやすく，2つに分けて書いておきました．

なお，プリントアウトを工夫して，表形式にする……つまり上部に見出しをつけて，以下はその内容だけを打出してゆくようにすれば，より見やすくなりますので，実用に際してはそれを考えるとよいでしょう．そのときは出力文字を右づめにするとより見やすくなります．PRINT USING文で「&　　&」を指定すると左づめになりますから，別の手法を考えなければいけません．

```
760   INPUT"ﾃｲｾｲｼﾀｲﾓﾉｦ ﾊﾞﾝｺﾞｳﾃﾞ ｼﾃｲｼﾃ ｸﾀﾞｻｲ   ﾄﾘｹｼ ﾅﾗ 999";TNUM
770   IF TNUM=999 THEN CLOSE:GOTO 630 '･･････････ﾃｲｾｲﾉ ﾊｼﾞﾒﾆﾓﾄﾞﾙ
780   IF TNUM<1 OR TNUM>8 THEN 760
790 ON TNUM GOSUB 830,840,850,860,870,880,890,900
800   PUT#1,N '･････････････････････････････････ﾌﾞﾌﾞﾝ ﾃｲｾｲｺﾞﾆ ｶｷｶｴﾙ
810   PRINT"1ｶｲﾉ ﾃｲｾｲﾊ 1ｺｳﾓｸﾃﾞｽﾉﾃﾞ ﾓﾄﾞﾘﾏｽ":CLOSE:GOTO 630
820 'ｼﾃｲﾉ ﾓﾉｦ ﾌﾞﾌﾞﾝﾃｷﾆ ﾃｲｾｲｽﾙ SUB
830   PRINT TITLE$(1);"? ";:LINE INPUT D1$:LSET RB$=D1$:RETURN
840   PRINT TITLE$(2);"? ";:LINE INPUT D2$:LSET RN$=D2$:RETURN
850   PRINT TITLE$(3);"? ";:LINE INPUT D3$:LSET RS$=D3$:RETURN
860   PRINT TITLE$(4);"? ";:LINE INPUT D4$:LSET RPG$=D4$:RETURN
870   PRINT TITLE$(5);"? ";:LINE INPUT D5$:LSET RH$=D5$:RETURN
880   PRINT TITLE$(6);"? ";:LINE INPUT D6$:LSET RPR$=D6$:RETURN
890   PRINT TITLE$(7);"? ";:LINE INPUT D7$:LSET RCO$=D7$:RETURN
900   PRINT TITLE$(8);"? ";:LINE INPUT D8$
910   IFD8$="" THEN D8$="ﾄｸﾆﾅｼ":           LSET RMEM$=D8$:RETURN
920 '
930 '*********** ﾃﾞｰﾀﾋｮｳｼﾞ SUB
940 PRINT USING"ﾊﾞﾝｺﾞｳ=&   &  1-ﾎﾝﾉﾀｲﾄﾙ=&                    &  2-ﾁｮｼｬ= &
    &";RNUM$;RB$;RN$
950   PRINT USING"3-ｼｭｯﾊﾟﾝｼｬ=&                 &       4-ﾍﾟｰｼﾞｽｳ=&    &     5-ﾊｯｺｳﾋﾞ=&
      &";RS$;RPG$;RH$
960   PRINT USING"6-ﾃｲｶ=&      &                    7-ﾎﾝﾉ ﾌﾞﾝﾙｲ=&         &";RPR$;R
CO$
970   PRINT"8-ﾒﾓ ";RMEM$
980 RETURN
990 '
1000 '*********** ﾊﾞﾝｺﾞｳ ﾌﾒｲﾉ ﾃｲｾｲ
1010 CLS
1020    FOR I=1 TO MAXNUM:GET#1,I
1030       GOSUB 940 '･･･････････････････････････････････ﾃﾞｰﾀﾋｮｳｼﾞ SUB
1040          IF I MOD 3=0 THEN 1050 ELSE 1080 '･･･････3ﾌﾞﾛｯｸﾃﾞ ｲﾁｼﾞﾃｲｼ
1050             PRINT"ﾊﾞﾝｺﾞｳｶﾞ ﾜｶｯﾀﾗ S ﾂｷﾞｦ ﾐﾙﾅﾗ ANY KEY PUSH"
1060             YN$=INKEY$:IF YN$="" THEN 1060
1070       IF YN$="S" OR YN$="s" THEN CLOSE:GOTO 620
1080     NEXT I
1090   PRINT"ﾃﾞｰﾀﾋｮｳｼﾞ ｵﾜﾘ｡ ﾊﾞﾝｺﾞｳｶﾞ ﾜｶｯﾀﾗ Y ﾓｳｲﾁﾄﾞﾐﾙﾅﾗ R ｦ ｵｽ"
1100   YN$=INKEY$:IF YN$="" THEN 1100
1110   IF YN$="Y" OR YN$="y" THEN CLOSE:GOTO 620 '･････ﾃｲｾｲﾉ ﾊｼﾞﾒﾆｺｸ
1120   IF YN$="R" OR YN$="r" THEN  1010 '･･･････････････ﾊﾞﾝｺﾞｳ ﾌﾒｲﾉ ﾃｲｾｲﾍ ﾓﾄﾞﾙ
1130 GOTO 1090
1140 '
1150 '========================= 4 PRINT OUT ==================================
1160 WIDTH 80,25:CLS
1170   GOSUB 1600 '･････････････････････････････････････OPEN SUB
1180   MAXNUM=LOF(1)
1190   PRINT"ｻｲﾀﾞｲｷﾛｸｽｳ=";MAXNUM:PRINT:CLOSE
1200   PRINT"1･････ｽﾍﾞﾃﾉ PRINT OUT   2･････ﾌﾞﾌﾞﾝﾉ PRINT OUT  3････ﾄﾘｹｼ"
```

※フローチャート：プログラムの流れを図に表わしたもの

なお，上の図は本プログラムの訂正部分のディテールフロー（詳細なフローチャート）です．プログラムと共に参照してください．

```
1210   PRINT"ﾄﾞﾚﾃﾞｽｶ ﾊﾞﾝｺﾞｳﾃﾞ ﾄﾞｳｿﾞ"
1220   POUT$=INKEY$:IF POUT$="" THEN 1220
1230   POUT=VAL(POUT$):IF POUT<1 OR POUT>3 THEN 1220
1240   IF POUT=3 THEN 150'ﾄﾘｹｼﾊ MENUﾍ
1250   GOSUB 1600 '.....................................OPEN SUB
1260   ON POUT GOSUB 1300,1350
1270 CLOSE:GOTO 150 '...............................MENUﾍﾓﾄﾞﾙ
1280 '********** ｽﾍﾞﾃﾉ PRINT OUT
1290         LPRINT SPACE$(30);"****** ﾜｶﾞﾔﾉ ｿﾞｳｼｮ *****":LPRINT
1300    FOR I=1 TO MAXNUM
1310      GET#1,I:GOSUB 1440 '.........................PRINTOUT SUB
1320    NEXT I
1330 RETURN
1340 '
1350   INPUT"ﾅﾝﾊﾞﾝｶﾗ ";STAR:IF STAR<1 OR STAR>MAXNUM THEN 1350
1360   INPUT"ﾅﾝﾊﾞﾝﾏﾃﾞ ";STP:IF STP<STAR OR STP>MAXNUM THEN 1360
1370    LPRINT SPACE$(30);"****** ﾜｶﾞﾔﾉ ｿﾞｳｼｮ *****":LPRINT
1380       FOR I=STAR TO STP
1390          GET#1,I:GOSUB 1440 '......................PRINT OUT SUB
1400       NEXT I
1410 RETURN
1420 '
1430 '********** PRINTOUT SUB
1440 LPRINT STRING$(79,"-")
1450   LPRINT USING"ﾊﾞﾝｺﾞｳ=&   &  1-ﾎﾝﾉﾀｲﾄﾙ=&                          &   2-ﾁｮｼｬ=&
     &";RNUM$;RB$;RN$
1460   LPRINT USING"3-ｼｭｯﾊﾟﾝｼｬ=&               &      4-ﾍﾟｰｼﾞｽｳ=&   &    5-ﾊｯｺｳﾋﾞ=
&        &";RS$;RPG$;RH$
1470   LPRINT USING"6-ﾃｲｶ=&     &                      7-ﾎﾝﾉ ﾌﾞﾝﾙｲ=&         &";RPR$
;RCO$
1480   LPRINT"8-ﾒﾓ ";RMEM$
1490 RETURN
1500 '
1510 '========================== 5 ｼｭｳﾘｮｳ==================================
1520 CLOSE:WIDTH 40,20:LOCATE 10,10:PRINT" THE END":END
1530 '=====================================================================
1540 '
1550 '********** DATA
1560 DATA "1      ﾎﾝﾉ ﾀｲﾄﾙ","2          ﾁｮｼｬﾒｲ","3      ｼｭｯﾊﾟﾝｼｬ","4        ﾍﾟｰｼﾞｽｳ
"
1570 DATA "5  ｵｸｻﾞｹ ﾊｯｺｳﾋﾞ","6           ﾃｲｶ","7      ﾎﾝﾉ ﾌﾞﾝﾙｲ","8 ﾒﾓ(150ﾓｼﾞｲﾅｲ)
"
1580 '
1590 '********* OPEN & FIELD SUB
1600 OPEN"R",#1,"0:BOOK"
1610 FIELD #1,5AS RNUM$,20AS RB$,15AS RN$,15AS RS$,5AS RPG$,10AS RH$,7AS RPR$,10
AS RCO$,150AS RMEM$
1620 RETURN
```

ランダムファイルとデータの削除

＊作成ずみのファイルから，データの一部を削除する

ファイルにデータを書きこみ，訂正をするときにはレコード番号を指定して，直接変更できるのがランダムファイルの利点です．いっぽう実際のデータを扱う時に，作成ずみのデータの中から不要なものを消したいことがあります．たとえば

レコード番号＝1　　データA
レコード番号＝2　　データB
レコード番号＝3　　データC

はすでに作成ずみで，この中からレコード番号の2番だけを削除して

レコード番号＝1　　データA
レコード番号＝2　　データC

というファイルにするにはどうしたらよいでしょうか？

＊＊配列にデータを入れれば処理はやさしいが……

ひとつの方法はこうです．削除したい番号をSとして，入力プログラムで求め（S＝2)，まず記録されているレコード番号の最大数を求めます．LOF関数により，これをMAXNとするといまの場合MAXN＝3になります．いっぽうデータを配列D(I)に代入することを考えて，これもあらかじめD(1)＝A，D(2)＝B，D(3)＝Cという処理をしておきます．これらの準備が終ったら

```
FOR  I=S  TO  MAXN-1 (この場合2TO2)
  D(I)=D(I+1) (D(2)=D(3)=Cになる)
NEXT  I
```

を実行すると，結果的にはD(1)＝A，D(2)＝C，D(3)＝Cになります．もとのファイルをKILLし，再度OPEN命令をして

```
FOR  I=1  TO  MAXN-1 (Iは1から2まで)
  PUT 「データ」(データをファイルに書きこむ)
NEXT  I
```

とすれば，データ数がひとつ減り，しかもレコード番号もひとつ減ったものが作成されるわけです．

データ数が少ないときはこのようにデータを一度配列へと送り，改めて記録しなおすことが可能です．しかし，データ数が多い時は，コンピュータが用意できる配列に限度があるため，実用になりません．

***2つのファイルを用意

もうひとつの方法は，ファイルを2つ用意してデータをいちど別のファイルに書きこむものです．この時もデータは最初

レコード番号＝1　　　データA
レコード番号＝2　　　データC
レコード番号＝3　　　データC

になり，これが別のファイルに送られます．もとのファイルを消し

```
FOR  I=1  TO  MAXN-1
  PUT 「データ」(ファイルにデータを書きこむ)
NEXT  I
```

というプログラムによって，もういちど新しく書きこみをすることで削除後のデータはレコード番号が1と2になります．この方法を使えば，データを配列に入れることなく，データの一部を削除して，しかもレコード番号はひとつ減ったものが作成されます．この方法はファイルを2つ用意しなければいけないこと，また，データを転送して書きこむので，読出しの回数が多くなるほど時間がかかることが欠点になりますが，ディスケットに納められる範囲のデータ数なら，コンピュータそのもののメモリには関係しないので多量のデータであっても削除が可能です．この方法をもう一度整理しましょう．

1 ファイルを2つ用意する
2 削除するレコード番号を求める
3 もとのファイルから削除指定のものを探し，データを詰める
4 別のファイルにデータを送る
5 もとのファイルからデータをすべて消し去る
6 別のファイルからデータを取出し，レコード番号をひとつだけ減らして，もとのファイルに書きこむ
7 別のファイルのデータをすべて消し去る

という作業により，データの一部が削除されるわけです．

※ランダムファイルでは，たとえば書きこみの時にPUT＃1として，レコード番号を省略する使いかたが許される．しかしレコード番号を指定したほうが，プログラミングが明快になるので，習慣として身につけたい

****削除プログラムの実例を考える

では，今の話をもとに，実際のプログラムで考えてみましょう．サンプルはとりあえず住所録ということで，氏名・住所・電話番号のデータをひとまとめにして1レコードに扱うことにします．データの新規作成は，プログラム中のDATA文で作ることにしましょう．ここではデータの追加と削除を使い手がコントロールします．

このプログラムはデータの新規作成・追加・削除の3つのメニュー

によって構成されています．新規作成はDATA文を読むだけで終るようになっていますし，追加はファイルの最大数を求め，それに＋1をしたものになるプログラムです．

つぎにデータの削除を説明しましょう．

*******GET＃1，PUT＃2でデータを転送**

まずファイルは2つ必要ですから，F-BASICを起動するとき，“How many files”の入力で，「2」を入力します．削除したいレコード番号はSAKUNUMとし（行番号640），この番号がファイルの最大数でなければ（途中にはさまれたデータなら）行番号680〜700で，削除したいもののあとに，削除したい番号以降のものを順に入れてデータをひとつずつ詰めます．

データを詰めるだけなら，ファイル番号は＃1だけを使えばいいだけで，GET＃1，I＋1：PUT＃1，Iで実現できます．つぎに＃2のファイルを開き，もとのデータのレコード番号よりひとつ少ないレコード番号で＃2へと転送します（行番号710〜760）．これで元のデータから，レコード番号がひとつだけ少ない，削除ずみのデータが＃2に収められました．つぎはもとのファイルをすべて消し去って（行番号780），そのあとに＃2から＃1へと転送します．

```
100 '================ ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ** ﾃﾞｰﾀﾉ ﾂｲｶ&ｻｸｼﾞｮ ** ===========
110 '
120 '                    **** HOWMANY FILES ﾃﾞ <<2>> ｦ ｵｽ
130 '
140 '=============================================================
150   DEFINT A-Z
200 WIDTH 40,20:CLS
210   PRINT"1･･･ｼﾝｷ ｻｸｾｲ":PRINT"2･･･ﾂｲｶ":PRINT"3･･･ｻｸｼﾞｮ"
220   PRINT:PRINT:INPUT"ﾄﾞﾚｦ ｴﾗﾋﾞﾏｽｶ";MENU
230   ON MENU GOSUB 300,400,600
240 GOTO 200
250 '
260 '********************* 1 ｼﾝｷ ｻｸｾｲ ******************************
300 GOSUB 1000 '･･････････････････････････････････OPEN #1 & FIELD SUB
310 N=1
320   READ D1$:IF D1$="END" THEN CLOSE:RETURN
330   READ D2$:READ D3$
340 GOSUB 1200 '･･････････････････････････････････LSET SUB
350   PUT#1,N:N=N+1:GOTO 320
360 '************************* 2 ﾂｲｶ ********************************
400 WIDTH 80,25:CLS
410 GOSUB 1000 '･･････････････････････････････････OPEN #1 & FIELD SUB
```

```
   1 ﾀﾅｶ ｼﾝｼﾞ          ﾐﾅﾄｸ ｱｻﾞﾌﾞ1-2-3                03-251-0050
   2 ﾔﾏﾀﾞ ｺｳｲﾁ         ｶﾅｶﾞﾜｸ ﾎﾝﾄﾞｵﾘ3-4-5             045-001-1050
   3 ｷｳﾁ ﾐﾄﾞﾘｺ         ｻｯﾎﾟﾛｼｵｵﾄﾞｵﾘ6-7-8             01-042-9999
   4 ｷﾀﾞ ﾖｼｺ           ｻｲﾀﾏｼﾐﾄﾞﾘﾁｮｳ3-1                0422-001-0909
ｻｸｼﾞｮ ｼﾀｲ ﾊﾞﾝｺﾞｳｦ ﾄﾞｳｿﾞ? 3
ｵﾜﾘ････E  ﾂﾂﾞｹﾙ････C ｦ ｵｽ ?
   1 ﾀﾅｶ ｼﾝｼﾞ          ﾐﾅﾄｸ ｱｻﾞﾌﾞ1-2-3                03-251-0050
   2 ﾔﾏﾀﾞ ｺｳｲﾁ         ｶﾅｶﾞﾜｸ ﾎﾝﾄﾞｵﾘ3-4-5             045-001-1050
   3 ｷﾀﾞ ﾖｼｺ           ｻｲﾀﾏｼﾐﾄﾞﾘﾁｮｳ3-1                0422-001-0909
ｻｸｼﾞｮ ｼﾀｲ ﾊﾞﾝｺﾞｳｦ ﾄﾞｳｿﾞ? 1
```

画面例▶

データが削除され，指定のものがもとのファイルから消えてレコード番号もおなじくひとつ減っていることを，上の画面ハードコピーのサンプルで見てください．

なお，削除したいものが最大レコード番号になるときは行番号800～830などのようにFOR～NEXTは使えませんので，転送ルーチンへ直接飛ばします（行番号660）．このプログラムでは，もとのファイルは“RFSAMPLE”，仮の形にして転送するファイルは“RFDUMMY”としています．

```
420 MAXNUM=LOF(1) '······························ｻｲﾀﾞｲﾚｺｰﾄﾞｽｳｦ ﾓﾄﾒﾙ
430 GOSUB 1400 '·································ﾃﾞｰﾀ ﾋｮｳｼﾞ SUB
440 N=MAXNUM+1
450   LINE INPUT"ﾅﾏｴ?     ";D1$
460   LINE INPUT"ｼﾞｭｳｼｮ? ";D2$
470   LINE INPUT"ﾃﾞﾝﾜ?    ";D3$
480  GOSUB 1200 '································LSET SUB
490 PRINT"ｺﾚﾃﾞ ﾋﾄﾘﾌﾞﾝ ｵﾜﾘ｡ ﾃｲｾｲ･･･T ﾂﾂﾞｹﾙ･･･C ｵﾜﾘ･･･E ｦ ｵｽ "
500   YN$=INPUT$(1)
510   IF YN$="T" OR YN$="t" OR YN$="ｶ" THEN 450
520   IF YN$="C" OR YN$="c" OR YN$="ｿ" THEN PUT #1,N:N=N+1:GOTO 450
530   IF YN$="E" OR YN$="e" OR YN$="ｲ" THEN PUT #1,N:CLOSE:RETURN
540 GOTO 490
550 '
560 '************************ 3 ｻｸｼﾞｮ ****************************
600 WIDTH 80,25:CLS
610 GOSUB 1000 '·······························OPEN#1 & FIELD SUB
620   MAXNUM=LOF(1)
630 GOSUB 1400 '·································ﾃﾞｰﾀﾋｮｳｼﾞ SUB
640   INPUT"ｻｸｼﾞｮ ｼﾀｲ ﾊﾞﾝｺﾞｳｦ ﾄﾞｳｿﾞ";SAKUNUM
650 IF SAKUNUM<1 OR SAKUNUM>MAXNUM THEN 640
660 IF SAKUNUM=MAXNUM THEN 710 '·················ｻｲｺﾞﾉﾃﾞｰﾀﾉ ｻｸｼﾞｮ
```

(次ページへ)

```
670 '---------- ﾃﾞｰﾀｦ ﾋﾄﾂ ﾂﾒﾙ
680  FOR I=SAKUNUM TO MAXNUM-1
690   GET#1,I+1:PUT#1,I
700  NEXT I
710 GOSUB 1100 '..................................OPEN #2 & FIELD SUB
720 '---------- #2ﾍ ﾃﾞｰｦ ﾃﾝｿｳ
730 FOR I=1 TO MAXNUM-1
740   GET#1,I:D1$=DAT1$:D2$=DAT2$:D3$=DAT3$
750   GOSUB 1300:PUT#2,I
760 NEXT
770 '
780 CLOSE#1:KILL"0:RFSAMPLE":GOSUB 1000 '.........ｲﾁﾄﾞ ﾌﾙｲﾃﾞｰﾀｦ ｹｽ
790 '---------- #1ﾍ ﾃﾞｰｦ ﾃﾝｿｳ
800 FOR I=1 TO MAXNUM-1
810   GET#2,I:D1$=DT1$:D2$=DT2$:D3$=DT3$
820   GOSUB 1200:PUT#1,I
830 NEXT
840 CLOSE#2:KILL"0:RFDUMMY" '.....................ﾀﾞﾐｰﾉ ﾌｧｲﾙｦ ｹｽ
850 '
860 PRINT"ｵﾜﾘ....E  ﾂﾂﾞｹﾙ....C ｦ ｵｽ ?":YN$=INPUT$(1)
870 IF YN$="E" OR YN$="e" OR YN$="ｲ" THEN CLOSE:RETURN
880 IF YN$="C" OR YN$="c" OR YN$="ｿ" THEN 620 ELSE 860
890 '
900 '=============== OPEN & FIELD SUB ====================
1000 OPEN"R",#1,"0:RFSAMPLE"
1010    FIELD#1,20AS DAT1$,30AS DAT2$,15AS DAT3$
1020 RETURN
1030 '
1100 OPEN"R",#2,"0:RFDUMMY"
1110    FIELD #2,20 AS DT1$,30AS DT2$,15AS DT3$
1120 RETURN
1130 '================= LSET SUB =========================
1200 LSET DAT1$=D1$:LSET DAT2$=D2$:LSET DAT3$=D3$:RETURN
1210 '
1300 LSET DT1$=D1$:LSET DT2$=D2$:LSET DT3$=D3$:RETURN
1310 '================= DATA ﾋｮｳｼﾞ SUB ===================
1400 FOR I=1 TO MAXNUM:GET#1,I
1410  PRINT USING"### &                  &   &
  &";I;DAT1$;DAT2$;DAT3$
1420 NEXT
1430 RETURN
1440 '
1450 '** DATA
1460 DATA ﾀﾅｶ ｼﾝｼﾞ, ﾐﾅﾄｸ ｱｻﾞﾌﾞ1-2-3,03-251-0050
1470 DATA ﾔﾏﾀﾞ ｺｳｲﾁ,ｶﾅｶﾞﾜｸ ﾎﾝﾄﾞｵﾘ3-4-5,045-001-1050
1480 DATA ｷｳﾁ ﾐﾄﾞﾘｺ,ｻｯﾎﾟﾛｼｵｵﾄﾞｵﾘ6-7-8,01-042-9999
1490 DATA ｷﾀﾞ ﾖｼｺ,ｻｲﾀﾏｼﾐﾄﾞﾘﾁｮｳ3-1,0422-001-0909
1500 DATA END
```

4

グラフィックスを

マスターしましょう

グラフィック座標とその処理

＊FM-77の座標と一般的な座標

F-BASICのグラフィック座標は多くのパーソナルコンピュータと同様に，左上を原点（ヨコ……X＝0，タテ……Y＝0）とし，左の図のようにプラス方向のみに設定されています．いっぽう私たちが数学で習ってきた平面座標は中心に原点があり，プラスとマイナスの双方へと広がっています．

画面にさまざまな模様を書いたりするグラフィックスでは座標の中心をどこに設定してもかまいません．しかし数学的なテーマを扱うときなどは，画面の中心に原点を設定したほうが実際の計算などと一致することもあって，理解しやすくなります．このための処理方法はいくつかありますが，これ以降のグラフィックスでは以下のルールを適用します．

○入力する数値などは画面の中心を原点とする一般的な座標とする
○入力後の，点の移動や回転変数なども同じ考えで処理する
○画面に出力するルーチンだけを座標変換する

つまりコンピュータが計算をする数値そのものはあくまでも数学的に取扱い，画面出力のときだけは変換してやります．この方式の利点は，グラフィックスの座標変換などの本質的な部分の計算はあくまでも数学的に正しい結果になることが挙げられます．欠点としてはプログラムが長くなること，画面出力用の変数が増えるために整理しにくくなることが挙げられます．しかし，計算は計算，表示は表示というように，ハッキリと分けたほうが混乱が少なくなりますし，デバッグするときも楽といえるでしょう．

＊＊原点の設定方法

原点を（X0, Y0）としたとき，入力されたX点，Y点を画面のどこに出力すればよいかというと，その式は

ヨコ位置（X方向）　X0＋X
タテ位置（Y方向）　Y0－Y

になります．F-BASICと一般座標の決定的な違いは，タテ方向で，方向の極性が反転してマイナスになります．また，タテ方向の寸法は0.45～0.4倍と，縮めないと見た目の寸法が狂います．縮率を手持ちのディスプレイに合わせましょう．

サンプルは5つの点を結ぶ図形を描き，指定の分だけ図形を移動させるものです．点を結ぶ図形で注意することは，書き始め（始点）からスタートして，つぎつぎと点を結んだとき，さいごの点（終点）を始点と一致させる，ということです．この処理をしないと閉じた図形になりません．ここでは配列変数をひとつ，余計に設計し，行番号200で終点（X(6)，Y(6)）にX(1)，Y(1)を代入しています．また，LINE文をサブルーチン化しているので，サブルーチンに引渡す変数をあらかじめ用意します(行番号250，370)．

※F-BASICの縮率は0.4495．CIRCLE文で正円を描き，ディスプレイの垂直振幅調整ツマミで正円に見えるようにしたら，その後の縮率係数も0.4495(≒0.45)にする

```
100 '=============== ｽﾞｹｲﾉ ｲﾄﾞｳ *5ﾂﾉ ﾃﾝｦ ﾑｽﾌﾞ =================
110 DIM X(6),Y(6),XM(6),YM(6)
120 WIDTH 80,25:CLS
130 SY=.43:X0=300:Y0=100:ZXS=0:ZXE=600:ZYS=0:ZYE=199
140 REM *SY=ｼｭｸﾘﾂ   X0,Y0=ｹﾞﾝﾃﾝ     ZXS,ZXE=Xｼﾞｸ:ZYS,ZYE=Yｼﾞｸﾉﾒﾓﾘ
150 '
160 FOR I=1 TO 5
170  READ X(I),Y(I)
180  DATA 10,10, 20,-30, 70,70, 0,120,  -50,-20,
190 NEXT I
200 X(6)=X(1):Y(6)=Y(1) '............ﾊｼﾞﾒﾄ ｵﾜﾘﾉ ﾃﾝｦ ｲｯﾁｻｾﾙ
210 '--------------------- ﾓﾄﾉ ｽﾞｹｲﾉ ﾋｮｳｼﾞ ----------------------
220 LINE(ZXS,Y0)-(ZXE,Y0),PSET,5:LINE(X0,ZYS)-(X0,ZYE),PSET,5
230 C=7 'ｼﾛｲ ｲﾛﾃﾞ ｴｶﾞｸ
240 FOR I=1 TO 5
250  XS=X0+X(I):YS=Y0-Y(I)*SY:XE=X0+X(I+1):YE=Y0-Y(I+1)*SY
260  GOSUB 430 '.....................LINE SUB
270 NEXT I
280 '----------------------- ｲﾄﾞｳ ﾆｭｳﾘｮｸ -----------------------
290 LOCATE 0,0:INPUT"xﾊ ｲｸﾂ ｲﾄﾞｳｻｾﾏｽｶ ";XM
300 LOCATE 0,1:INPUT"yﾊ ｲｸﾂ ｲﾄﾞｳｻｾﾏｽｶ ";YM
310 FOR I=1 TO 6
320   XM(I)=X(I)+XM:YM(I)=Y(I)+YM '......ｲﾄﾞｳﾌﾞﾝｦ ｸﾜｴﾙ
330 NEXT
340 '------------------ ｲﾄﾞｳ ｽﾞｹｲﾉ ﾋｮｳｼﾞ ----------------------
350 C=6 'ｷｲﾛｲ ｲﾛﾆｽﾙ
360 FOR I=1 TO 5
370  XS=X0+XM(I):YS=Y0-YM(I)*SY:XE=X0+XM(I+1):YE=Y0-YM(I+1)*SY
380  GOSUB 430 '.....................LINE SUB
390 NEXT I
400 END
410 '
420 '**** LINE SUB ****
430 LINE(XS,YS)-(XE,YE),PSET,C
440 RETURN
```

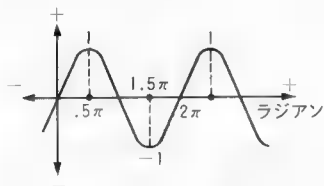
三角関数と円の描きかた

＊応用範囲の広い SIN, COS

三角関数は古くは土地測量などから起った考え方で，辺の長さと，それが作る角度などを知ることにより，他の辺の長さを求めたりする方法としても知られています．また物体や電気の振動などとも関係のあるもので広く使われます．

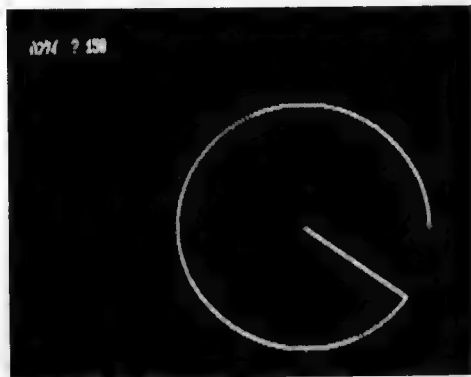
コンピュータにおける三角関数も，数学・物理・電気の分野などで応用されるほか，グラフィックスとも深い関係にあります．ここではまず，サイン（SIN），コサイン（COS）とラジアン値，角度の関係，それらをもとにして，角度を入力すると三角関数の値を画面に点としてそれを出力するプログラムを作り，動かしながら理解してゆきましょう．サンプルを見てください．

```
100 '============= ｻﾝｶｸ ｶﾝｽｳ  SIN COS / ﾌﾟﾛｯﾄ ==========================
110 '
120 PI=3.14159265#:DEGRAD=PI/180:X0=200:Y0=100
130 WIDTH 80,25:CLS
140 'ﾏｴｼｮﾘ
150 LOCATE 0,24:COLOR 5:PRINT"-- PLOT SIN  ";:COLOR 6:PRINT"-- PLOT COS"
160 COLOR 7
170 LINE(X0,Y0-100)-(X0,Y0+100),PSET,7 '·················Yｼﾞｸ
180 LINE(X0-50,Y0)-(X0+360,Y0),PSET,7 '··················Xｼﾞｸ
190 'ﾆｭｳﾘｮｸ
200 LOCATE 0,0:INPUT"ｶｸﾄﾞ  ";DEG:RAD=DEG*DEGRAD
210 LOCATE 0,1:PRINT"ﾗｼﾞｱﾝﾁ=";RAD
220 LOCATE 0,2:PRINT USING"SIN=#.##   COS=#.##";SIN(RAD);COS(RAD)
230 GOSUB 260:GOTO 200
240 '
250 '--------------PSET SUB ----------------
260 X=X0+DEG:Y1=Y0-70*SIN(RAD) '··············PLOTｽﾙ ｲﾁﾉ ｹｲｻﾝ(SIN)
270           Y2=Y0-70*COS(RAD) '··············PLOTｽﾙ ｲﾁﾉｹｲｻﾝ(COS)
280 'ﾌﾞﾘﾝｷﾝｸﾞPSET
290 FOR I=1 TO 40
300   PRESET(X,Y1):PRESET(X,Y2):FOR T=0 TO 100:NEXT
310   PSET(X,Y1,5):PSET(X,Y2,6):IF I MOD 5=0 THEN BEEP
320 NEXT I
330 '
340 LOCATE 0,3:PRINT"OK? PUSH ANY KEY"
350 WHILE INKEY$="":WEND
360 FOR E=0 TO 3:LOCATE 0,E:PRINT SPACE$(25);:NEXT '·····ﾓｼﾞ ｼｮｳｷｮ
370 RETURN
```

**三角関数をプロットしてゆく

このプログラムでは，まず原点になる点をX0＝200，Y0＝100に設定し，そこからX軸・Y軸の2本を作ります（行番号170，180）．角度を入力すると，それが何ラジアンになるかを計算させ（行番号200），そのSIN値・COS値を出力します（行番号220）．このあとにプロットをするサブルーチンを呼んでいますが，PSET文はXとYの2つの位置を指定するので，多少の工夫をしなければいけません．まずX方向ですが，角度は一般に0～360°の間の値をとります．したがって1°につきX方向は1ドット分で対応させればいいでしょう．たとえば90°という入力があった時は，X方向はX0＋90＝200＋90＝290になります．

いっぽう，Y方向はSIN，COSの値になるのですが，双方とも角度がたとえ－100°であっても600°であっても，最大で1，最小で－1にしかなりません．したがって結果を50倍～100倍してやらないとPSET指定できないのです．このプログラムでは70倍していますから，最大でY0＋70＝170，最小でY0－70＝30という値になります（行番号260，270）．なお，ここではディスプレイのタテ方向の補正値は使っていません（本質的に不要です）．

このプログラムではPSETを何回でも重ねてできるように作っているので，入力角度にしたがってどこに点が打たれるかがよく分るようにブリンキング（点滅）をさせています．それが行番号290～320の文です．PSETとPRESETをくり返し，チカチカと光らせてから，正式に点灯させると共に，BEEP(ビープ)音でそれを知らせるようにしました．入力を何回もくり返すと，やがてサイン波形とコサイン波形に近づいてゆきます．

▲SINの例

▲円を描くプログラムの実行途中の画面

▶SIN, COSを使って円を描くプログラム

```
100 '******* SIN COS ト エン *********
110 PI=3.14159:X0=320:Y0=100:SY=.43
120 WIDTH 80,25:CLS
130 '
140 INPUT"ﾊﾝｹｲ   ";R
150 '
160 FOR RAD=0 TO 2*PI STEP PI/360
170   X=R*COS(RAD):Y=R*SIN(RAD)
180     X=X0+X:Y=Y0-Y*SY
190     LINE(X0,Y0)-(X,Y),PSET,5
200     LINE(X0,Y0)-(X,Y),PRESET
210   PSET(X,Y,7):'ｴﾝｼｭｳｦ ｴｶﾞｸ
220 NEXT
230 END
```

多角形の描きかた

＊円は多角形？

前のページの円の描き方のプログラムを見ても分るように，円の半径をRとしたとき，そこから左まわりにある角度dだけ移動した円周上の点は

$X=R\times\cos(d)$　　　$Y=R\times\sin(d)$

になります．角度を小さくして移動すると，点の軌跡が円になるわけですが，円周上の点を，たとえば120°ステップにすると，下の図のAのように，正三角の頂点を求めることができます．この頂点を結べば正三角形が描けますが，図でも分るように左に傾いたものになります．

いっぽう図Bのように，円の中心から一辺の長さR，角度は0°の位置をもとめると，そこは点Eになります．もうひとつ，円の中心から120°の点の位置を求めると，それは②の点Eになりますし，240°の点の位置は③の点Eになります．こうすれば3つの線分を求めることができるのですが，このままでは三角形になりません．

そこで図B′のように

①の線分をまず描く，つぎに①のEを線の描きはじめとして②の線分を描く，そのつぎに②のEを線の描きはじめとして③を描く……

というようにすれば，見なれた形の三角形になります．

右ページのプログラムはこうして多角形を描くもので，行番号230で，CONNECT命令で線分を描いたら，行番号240でスタート座標値（XS，YS）をエンド座標に変えています．

A)

B)

B′)

多角形プログラムの実行例▶

```
100 '************** ｱﾀｴﾗﾚﾀﾍﾝﾉﾅｶﾞｻﾃﾞ Nｶｸｹｲｦ ｶｸ ****************
110 '
120 ON ERROR GOTO 330 '......................ｴﾗｰ ｼｮﾘｮｳ
130    WIDTH 80,25
140    PI=3.14159:SY=.43
150 'ﾆｭｳﾘｮｸ
160 CLS:LOCATE 10,0:INPUT"ﾅﾝｶｸｹｲ ｦ ﾂｸﾘﾏｽｶ ";N%
170   IF N%<3 THEN BEEP:GOTO 160
180   LOCATE 50,0:INPUT"ﾍﾝﾉ ﾅｶﾞｻ(50-150)  ";HEN
190 'ﾀｶｸｹｲｦ ｴｶﾞｸ
200   XS=250:YS=190:CLS '....................XS,YS=ｼｭｯﾊﾟﾂﾃﾝ
210 FOR RAD=0 TO 2*PI+.1 STEP (2*PI)/N%
220    XE=XS+HEN*COS(RAD):YE=YS-HEN*SIN(RAD)*SY
230        CONNECT(XS,YS)-(XE,YE),6 '....2ﾃﾝｦ ﾑｽﾌﾞ
240    XS=XE:YS=YE '.......................ｶｷﾊｼﾞﾒﾉ ｲﾁｦ ｶｴﾙ
250 NEXT RAD
260 'ｴｶﾞｲﾀ ｽﾞｹｲﾉ ｾﾂﾒｲﾄ ﾂｷﾞﾉ ﾆｭｳﾘｮｸ
270   LOCATE 20,0:PRINT USING"##ｶｸｹｲ     ﾍﾝﾉ ﾅｶﾞｻ=###";N%;HEN
280   LOCATE 20,2:PRINT"ﾂｷﾞｦ ﾂｸﾙﾅﾗ ANY KEY HIT !"
290      WHILE INKEY$="":WEND
300 GOTO 160
310 '
320 'ｴﾗｰｼｮﾘ
330 LOCATE 20,20:PRINT"ﾍﾝｦ ﾐｼﾞｶｸ":FOR T=0 TO 2000:NEXT
340 CLS:RESUME 180
```

配列と行列の和・積

*行列とは?

物理における力の大きさと力の方向などの，要素（成分）をひとまとめにして表わしたりすると，たとえばそれをいろいろと変換する時に便利でもあり，説明あるいは理解しやすくなります．このような時に使うのが行列（マトリクス：matrix）と呼ばれるものです．

行列は物理や幾何学その他において重要な働きをするもので，これだけを専門に勉強しても勉強しきれないものといえます．本節ではこの行列とその計算（演算）の基礎について触れることにします．これから先に出てくる3次元図形その他を本格的にマスターする時，マトリクスの演算が関係してきますが，その予備編と考えていただいてもけっこうです．

**行列の和と積のもとめかた

行列は読んで字のごとし…行（上から下へと並ぶ：column（コラム））と列(左から右へと並ぶ：row（ロウ））のそれぞれに成分があるもので，たとえば2行3列の行列（マトリクス）では合計6個の成分から構成されます．コンピュータでは，たとえばそれを$M(2, 3)$というようにし，最初の添字は行，つぎの添字で列を表わします．

行列の計算ですが，たとえば$M1$，$M2$の2つがあった時，加減算については，それぞれの位置にある成分を加減するだけの作業になり，ごく簡単です．たとえば

$$\begin{pmatrix} 2, & 1, & 3 \\ -1, & 5, & 0 \end{pmatrix} + \begin{pmatrix} -1, & 2, & 4 \\ 3, & 8, & 5 \end{pmatrix} = \begin{pmatrix} 1, & 3, & 7 \\ 2, & 13, & 5 \end{pmatrix}$$

加算のプログラムとあわせて見てください．

いっぽう行列の乗算（かけ算）は特別で，たとえば$M1=m$行l列，$M2=l$行n列のとき

$$M_{ij}=\sum_{k=1}^{l} M1_{ik} \times M2_{kj} \quad (i=1, 2, \cdots m, j=1, 2, \cdots n)$$

Σはインテグレーション(合算，まとめる)で$M1$と$M2$の成分を式にしたがって乗算してゆき，それを合計したものになります．

$$\begin{pmatrix} 2, & 4 \\ 3, & 5 \end{pmatrix} \times \begin{pmatrix} 3, & 1, & -1 \\ 6, & 8, & 2 \end{pmatrix}$$

では，$M1$が2行2列（$m=2$，$l=2$），$M2$が2行3列（$l=2$，$n=$

```
100 'MATRIX 1 ** ｶｹﾞﾝｻﾞﾝ
110 WIDTH 40,20:CLS
120 DIM M1(2,3),M2(2,3),M(2,3)
130 '*** M1ﾉ ﾖﾐｺﾐ
140 FOR G=1 TO 2 'ｷﾞｮｳ
150   FOR R=1 TO 3 'ﾚﾂ
160     READ M1(G,R)
170   DATA 2,1,3,  -1,5,0
180 NEXT R,G
190 '*** M2ﾉ ﾖﾐｺﾐ
200 FOR G=1 TO 2
210   FOR R=1 TO 3
220     READ M2(G,R)
230   DATA -1,2,4,  3,8,5
240 NEXT R,G
250 '*** ｴﾝｻﾞﾝ
260 FOR G=1 TO 2
270   FOR R=1 TO 3
280   M(G,R)=M1(G,R)+M2(G,R)
290 NEXT R,G
300 '*** ﾃﾞｨｽﾌﾟﾚｲ
310 FOR G=1 TO 2
320   FOR R=1 TO 3
330      PRINT M(G,R);
340   NEXT R:PRINT
350 NEXT G
360 END
```

▲行列の和をもとめる

```
100 'MATRIX 2 ** ｼﾞｮｳｻﾞﾝ
110 WIDTH 40,20:CLS
120 DIM M1(2,2),M2(2,3),M(2,3)
130 '*** M1ﾉ ﾖﾐｺﾐ
140 FOR G=1 TO 2 'ｷﾞｮｳ
150   FOR R=1 TO 2 'ﾚﾂ
160     READ M1(G,R)
170   DATA 2,4,  3,5
180 NEXT R,G
190 '*** M2ﾉ ﾖﾐｺﾐ
200 FOR G=1 TO 2
210   FOR R=1 TO 3
220     READ M2(G,R)
230   DATA 3,1,-1,  6,8,2
240 NEXT R,G
250 '*** ｴﾝｻﾞﾝ
260 FOR R=1 TO 3
270   FOR G=1 TO 2
280 FOR K=1 TO 2 'ｲﾝﾃｸﾞﾚｰｼｮﾝ
290   M(G,R)=M1(G,K)*M2(K,R)+M(G,R)
300 NEXT K,G,R
310 '*** ﾃﾞｨｽﾌﾟﾚｲ
320 FOR G=1 TO 2
330   FOR R=1 TO 3
340      PRINT USING"### ";M(G,R);
350   NEXT R:PRINT
360 NEXT G
370 END
```

▲行列の積をもとめる

3）ですから，この結果のMは$M(i,j)=M(2,3)$になります．たとえば$M(2,3)$はkを1からl（=2）まで変え

$k=1$ ：$M1(\overset{i}{2},\ \overset{k}{1})\times M2(\overset{k}{1},\ \overset{j}{3})=3\times(-1)=-3$

$k=2$ ：$M1(\overset{i}{2},\ \overset{k}{2})\times M2(\overset{k}{2},\ \overset{j}{3})=5\times 2=10$

を合計したものですから，$M(2,3)=7$になります．

2つの行列の乗算のプログラム列を上に示します．FOR～NEXTのループがひとつ追加され，前に述べた式にしたがってインテグレーションをしているわけです．

さきほどの$M1\times M2$は

$$\begin{pmatrix}30, & 34, & 6\\ 39, & 43, & 7\end{pmatrix}$$

になります．

平面図形の回転と行列

＊回転を表わすマトリクス

平面図形にある点は２つの位置要素…ヨコ位置を表わすものとタテ位置を表わすもの，一般には x と y が使われる…で扱うことができます．点 (x, y) が原点を中心にしてある角度 d だけ回転した時の新しい点 (x', y') は次の式で表わされます．

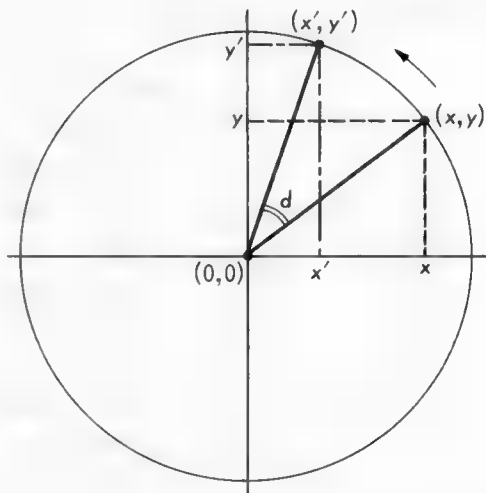
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos d & -\sin d \\ \sin d & \cos d \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix}$$

つまり回転後の x', y' は行列演算により計算できるのです．さっそくこれをプログラミングしてみましょう．前に説明した行列の乗算がそのままあてはめられるので，とりあえずそれを忠実に守ったプログラムにします．$M1$は２行２列（$m=2$，$l=2$），$M2$は２行１列（$l=2$，$n=1$）なので，この結果は２行１列になります．

演算は

$$M_i = \sum_{k=1}^{l} M1_{ik} \times M2_k \quad (i = 1 \cdots 2)$$

になります．jに関係する項は出てきません．また回転は反時計まわりになります．このプログラムによって求めたx', y'が正しいかどうか

```
100 'ﾃﾝﾉ ｶｲﾃﾝﾄ MATRIX
110 DEFDBL D
120 DIM M1(2,2),M2(2),M(2)
130 'ﾓﾄﾉｲﾁﾉ ﾆｭｳﾘｮｸ
140 INPUT"ﾃﾝxﾉ ｲﾁ ";M2(1)
150 INPUT"ﾃﾝyﾉ ｲﾁ ",M2(2)
160 'ｶｲﾃﾝｶｸﾉ ﾆｭｳﾘｮｸ
170 INPUT"ﾅﾝﾄﾞ ｶｲﾃﾝ ｻｾﾏｽｶ  ";D
180 D=(3.14159265359#/180)*D
190 M1(1,1)=COS(D):M1(1,2)=-SIN(D)
200 M1(2,1)=SIN(D):M1(2,2)=COS(D)
210 'ｴﾝｻﾞﾝ
220 FOR I=1 TO 2
230  FOR K=1 TO 2
240   M(I)=M1(I,K)*M2(K)+M(I)
250  NEXT K,I
260 '
270 PRINT"x'=";M(1)
280 PRINT"y'=";M(2)
```

は，方眼紙とコンパス，それに分度器を用意して実際に試してみるとよいでしょう．コンパスの針の位置が$x=0$，$y=0$になりますから，そこから半径10cmほどの大きな円を描き，方眼紙に目盛を打つことで確認できます．なお，このプログラムでは角度についてのみ倍精度を使っています．単精度では計算値に誤差が出ます（例：$x=10$，$y=0$として，90度回転させると$x'=0$，$y'=10$になるのですが，コンピュータの結果はこれとは一致しません）．また，F-BASICでは角度はラジアン値を使うので，角度で扱う時はあらかじめそれをラジアン値に変換しておくことが必要です．

**ユーザー定義関数を使ってみる

点の回転位置は以上のように行列を使った変換式で求められますが実際には

$$x'=x\cos d-y\sin d$$
$$y'=x\sin d+y\cos d$$

とおなじことになります．プログラミング上はこちらのほうがてっとり早いので，この式を使って，回転後の位置を計算させる方法が，よく使われます．また，F-BASICはユーザー定義関数が使えるので，それを応用すると，とてもスッキリとしたプログラムになります．そのサンプルプログラムを見てください．

新しい点をXR，YRとすれば，これは，いずれも（X，Y，D）という要素をパラメータとするものになるので，サンプルのように，行番号120，130でそれぞれを定義しておけば，FM-77が内部で自動的に計算をしてくれることになります．この回転数の点の位置を求めることを基本にして，グラフィック命令と組合せることによりいろいろな図形が描けることになります．

```
100 'ﾃﾝﾉｶｲﾃﾝﾄ DEF FN
110 DEFDBL D
120 DEF FNXR(D,X,Y)=X*COS(D)-Y*SIN(D)
130 DEF FNYR(D,X,Y)=X*SIN(D)+Y*COS(D)
140 '
150 INPUT"ﾃﾝxﾉ ｲﾁ ";X
160 INPUT"ﾃﾝyﾉ ｲﾁ ";Y
170 INPUT"ﾅﾝﾄﾞ ｶｲﾃﾝ ｻｾﾏｽｶ  ";D
180 '
190 D=(3.14159265359#/180)*D
200 PRINT"x'=";FNXR(D,X,Y)
210 PRINT"y'=";FNYR(D,X,Y)
```

DEF FNを使った例▶

```
ﾃﾝxﾉ ｲﾁ ? 10
ﾃﾝyﾉ ｲﾁ 0
ﾅﾝﾄﾞ ｶｲﾃﾝ ｻｾﾏｽｶ  ? 135
x'=-7.07107
y'= 7.07107
```

▲実行結果の例（どちらのプログラムでも同じ）

原点を中心にして図形を回転させる

*回転させる前に

図形を回転させる，というとひどくむずかしいように思えますが，はじめに説明したように，原点を中心にして反時計まわりにある角度だけ回転させたときのX，Y座標の変換式を使いさえすれば，回転後の座標値XR，YRは簡単に求めることができます．ここでは前出の5つの点を結んで作る図形をそのまま使い，それを指定の角度だけ回転させましょう．

※π（パイ）＝円周率．円における直径と円周の比率で3.141592653589793

なお，F-BASICでは角度にラジアン値を使います．私たちが小さい頃から学んできた角度は，コンパスをグルリとひとまわしすると360°になるものでした．これに対し，ラジアン値では2πになります．つまり360°＝2π，180°＝π，90°＝0.5πといった対応になります．したがって「何度回転させるか」という時も，角度ではなく，それをラジアン値にしなければいけないのです．1°に相当するラジアン値は（2π÷360）になります．なお，BASICプログラムではこれを（π÷180）にし，「3.14159／180」とか，あらかじめπをPIという変数に代入し，「PI＝3.14159」としておき，たとえば「PI／180」などとするのが普通です（ここではDEGRAD）．

**回転プログラムの作りかた

回転のプログラムの基本は

○元になる図形の点（X，Y）を求め，配列に入れる

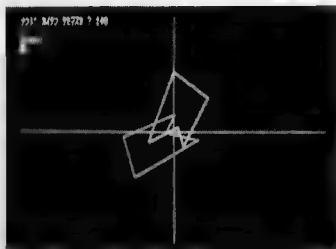
○何度回転させるかを求め，その角度をラジアン値に変換する

○回転後の座標値がどうなるかを計算させ，その結果を別の配列に入れる（XR，YR）

○グラフィック表示のサブルーチンを呼び出し，回転した図形を描かせる

ということになります．サンプルプログラムではFNXR，FNYRという，2つのユーザー定義関数を作り，行番号340でその関数を呼び出して計算させ，XR(I)，YR(I)という配列にその結果を代入させています．パラメータはD，X，Yの3つですが，Dについては行番号320で求めています．X，Yについては，X(I)とY(I)を配列から呼び出して，それをX，Yに代入しています．

▲回転図形の例

また原点からX方向，Y方向への座標軸は，ZXS，ZYEなどとこれ

```
100 '=============== ｽﾞｹｲﾉ ｶｲﾃﾝ *5ﾂﾉ ﾃﾝｦ ﾑｽﾌﾞ ==================
110 DEF FNXR(D,X,Y)=X*COS(D)-Y*SIN(D)
120 DEF FNYR(D,X,Y)=X*SIN(D)+Y*COS(D)
130 DIM X(6),Y(6),XR(6),YR(6)
140 WIDTH 80,25:CLS
150 DEGRAD=3.14159/180
160 SY=.43:X0=300:Y0=100:ZXS=0:ZXE=600:ZYS=0:ZYE=199
170 REM *SY=ｼｭｸﾘﾂ   X0,Y0=ｹﾞﾝﾃﾝ     ZXS,ZXE=Xｼﾞｸ:ZYS,ZYE=Yｼﾞｸﾉﾒﾓﾘ
180 '
190 FOR I=1 TO 5
200  READ X(I),Y(I)
210  DATA 10,10, 20,-30, 70,70, 0,120,  -50,-20,
220 NEXT I
230 X(6)=X(1):Y(6)=Y(1) '............ﾊｼﾞﾒﾄ ｵﾜﾘﾉ ﾃﾝｦ ｲｯﾁｻｾﾙ
240 '-------------------- ﾓﾄﾉ ｽﾞｹｲﾉ ﾋｮｳｼﾞ ----------------------
250 LINE(ZXS,Y0)-(ZXE,Y0),PSET,5:LINE(X0,ZYS)-(X0,ZYE),PSET,5
260 C=7 'ｼﾛｲ ｲﾛﾃﾞ ｴｶﾞｸ
270 FOR I=1 TO 5
280  XS=X0+X(I):YS=Y0-Y(I)*SY:XE=X0+X(I+1):YE=Y0-Y(I+1)*SY
290  GOSUB 450 '......................LINE SUB
300 NEXT I
310 '---------------------- ｶｲﾃﾝ ﾆｭｳﾘｮｸ -----------------------
320 LOCATE 0,0:INPUT"ﾅﾝﾄﾞ ｶｲﾃﾝ ｻｾﾏｽｶ ";D:D=D*DEGRAD 'ﾗｼﾞｱﾝ ﾍﾝｶﾝ
330 FOR I=1 TO 6
340   X=X(I):Y=Y(I):XR(I)=FNXR(D,X,Y):YR(I)=FNYR(D,X,Y)
350 NEXT
360 '------------------ ｶｲﾃﾝ ｽﾞｹｲﾉ ﾋｮｳｼﾞ ----------------------
370 C=6 'ｷｲﾛｲ ｲﾛﾆｽﾙ
380 FOR I=1 TO 5
390  XS=X0+XR(I):YS=Y0-YR(I)*SY:XE=X0+XR(I+1):YE=Y0-YR(I+1)*SY
400  GOSUB 450 '.....................LINE SUB
410 NEXT I
420 END
430 '
440 '**** LINE SUB ****
450 LINE(XS,YS)-(XE,YE),PSET,C
460 RETURN
```

も変数化しています．行番号250で座標軸を描いていますが，あらかじめ変数化しておけば，変更をする時もプログラム中の実行ルーチン（ここでは行番号250）そのものに手を加えず，行番号160の初期値設定だけに手を加えればよいので，修正が楽になります．変数の種類が増えてくると管理がたいへんになりますが，REM文などで変数リストを作成しておけば，忘れてしまってもこの変数リストを見ることでフォローできます．

図形の連続回転

＊菱形を回転させてみる

つぎは基本図形を一定の角度でつぎつぎと回転させてみましょう．このプログラムで使った図形は菱形で4つの点は(30, 0)，(100, −30)，(170, 0)，(100, 30) になります．回転角度は1回につき 12° とし，これを30回くりかえすとちょうど360°になります．まず基本の点をX，Yとして，配列に読みこませます（行番号220〜230)．つぎに回転後の30個の座標をXR，YRとして2次元配列を用意し，まずこの配列に回転の計算結果を入れてしまいます．この2次元配列はたとえば

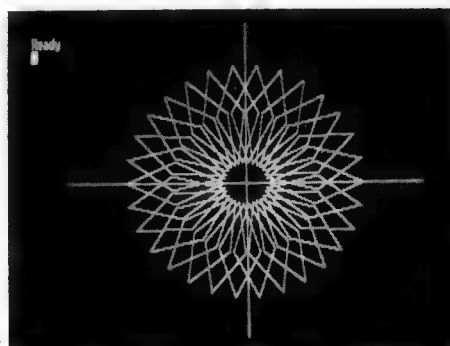
XR（1，1）＝12°左へ回転した時の，元はx＝30のX座標
XR（1，2）＝12°左へ回転した時の，元はx＝100のX座標
XR（1，3）＝12°左へ回転した時の，元はx＝170のX座標
XR（1，4）＝12°左へ回転した時の，元はx＝100のX座標
XR（1，5）はXR（1，1）とおなじX座標

ということです．添字の最初はそれぞれの回転角度に対応し，つぎの添字は元のX座標の個数に対応しています．

＊＊計算しながら図形を描くか，計算してからにするか？

回転した時の座標を求め，それに応じて図形を描くとき，このプログラムでは一括してまず計算をさせ，それを配列にしまっています．その後にまとめて図形を描くので，描くスピードは速くなります．

いっぽう1回の計算をしたらひとつの図形を描く方法も考えられますが，この時は原則として配列のような，回転後の位置をメモリしておく手段は不要です．回転後の座標を求めたら図形を描き，変数に入っている座標値はつぎつぎと捨てていってもかまいません．その代りひとつの図形を描いてはまた計算をするので，図形を描いたあとに少し時間がかかり，見かけ上ゆっくりと30個の図形を描きます．

どちらの方法を採用するのかはその目的によって選べばよいのですが，人間の心理としては図がつぎつぎと早く描かれたほうがなんとなく気持のよいものです．このプログラムでは計算中の待ち時間に，それを見ている人間が「いったい，いつ図を描き始めるのだろう？」とイライラしないように行番号250，280でメッセージを出して，あとどのくらい待てばよいかを知らせています．単に「待ってください」とするよりも，行番号280のように残り回数を知らせる方が効果的です．

回転する菱形▶

```
100 '==================== ｽﾞｹｲﾉ ｶｲﾃﾝ *ﾋｼｶﾞﾀﾉ ｶｲﾃﾝ =====================
110 DEFINT I,J
120 DEF FNXR(D,X,Y)=X*COS(D)-Y*SIN(D)
130 DEF FNYR(D,X,Y)=X*SIN(D)+Y*COS(D)
140 '
150 DIM X(5),Y(5),XR(30,5),YR(30,5)
160 DEGRAD=3.14159/180
170 SY=.43:X0=300:Y0=100:ZXS=50:ZXE=550:ZYS=0:ZYE=199
180 REM *SY=ｼｭｸﾘﾂ   X0,Y0=ｹﾞﾝﾃﾝ     ZXS,ZXE=Xｼﾞｸ:ZYS,ZYE=Yｼﾞｸｦ
190 '
200 WIDTH 80,25:CLS
210 '-------------------------- 4ﾂﾉ ﾃﾝﾉ ﾖﾐｺﾐ ----------------------------
220 FOR I=1 TO 4:READ X(I),Y(I):NEXT I
230  DATA 30,0,  100,-30,  170,0,  100,30
240 X(5)=X(1):Y(5)=Y(1) '................ｼｭｳﾃﾝｦ ｼﾃﾝﾄ ｵﾅｼﾞﾆｽﾙ
250 LOCATE 20,10:PRINT"ﾀﾀﾞｲﾏ ｹｲｻﾝﾁｭｳ ｼﾊﾞﾗｸ ｵﾏﾁｸﾀﾞｻｲ"
260 '--------------------------- ｶｲﾃﾝ ｹｲｻﾝ -------------------------------
270 FOR I=1 TO 30 '.....................30ｶｲ ｶｲﾃﾝｻｾﾙ
280     LOCATE 30,13:PRINT (30-I) '.....ｹｲｻﾝﾉ ﾉｺﾘﾉ ｶｲｽｳ ﾋｮｳｼﾞ
290  D=D+12*DEGRAD '....................ｶｸﾄﾞｦ 12°ｽﾞﾂ ﾌﾔｽ
300   FOR J=1 TO 5 '....................ﾋﾄﾂﾉ ｽﾞｹｲﾉ ｻﾞﾋｮｳ ｹｲｻﾝ
310      X=X(J):Y=Y(J):XR(I,J)=FNXR(D,X,Y):YR(I,J)=FNYR(D,X,Y)
320 NEXT J,I
330 '----------------------- X・Y ｼﾞｸ ｦ ｴｶﾞｸ ----------------------------
340 CLS
350 LINE(ZXS,Y0)-(ZXE,Y0),PSET,5:LINE(X0,ZYS)-(X0,ZYE),PSET,5
360 '--------------------- ｶｲﾃﾝ ｽﾞｹｲｦ ｴｶﾞｸ ------------------------------
370 FOR I=1 TO 30 '.....................30ｶｲ ｴｶﾞｸ
380   FOR J=1 TO 4 '....................ﾋﾄﾂﾉ ｽﾞｹｲｦ ｴｶﾞｸ
390   LINE(X0+XR(I,J),Y0-YR(I,J)*SY)-(X0+XR(I,J+1),Y0-YR(I,J+1)*SY),PSET,6
400 NEXT J,I
410 '
420 END
```

3次元図形の基礎

＊3次元図形とその表現

立体物をそれらしく見えるように描く工夫は，多くの芸術家や科学者が古くから研究してきたことです．現在の，レーザー光線を使って立体物を空間に投影するような特殊な技術を使うことを別にすれば，立体物はすべて平面……紙やディスプレイなどに投影されます．そして平面に描かれたものに陰影をつけたり，手前にあるものと奥にあるものとの大きさを変えたりして表現するのが一般的です．さらに立体物の表現は，工業的な目的に使うのか，芸術的な表現にするのかなど，そのねらいによってデフォルメされることになります．

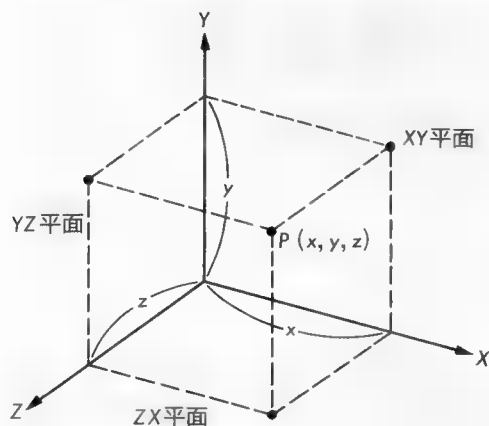
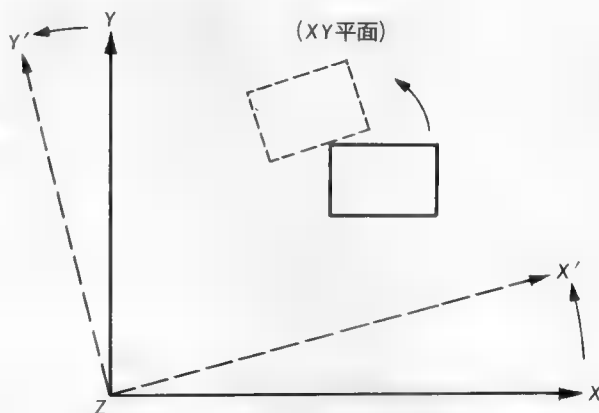
パーソナルコンピュータによる立体物の表現（以下3次元図形）もそのねらいが何であるかによって，処理の内容が異なります．ここでは芸術的表現はさておいて，ごくやさしい3次元図形を考えることにしましょう．記憶容量が大きく，高速演算が可能なコンピュータの場合は，ディスプレイの1点1点について3次元的な処理が可能になりますが，そうでない場合は限度があることも理由のひとつです．

＊＊遠くにあるものは小さく見える

3次元図形を考えるときの第一の原則は，遠くにあるものは小さく見えるため，平面に写しとる時も小さくなる，ということです．もうひとつは自分が立っている位置と，対象物との距離によっても見えかたが異なることや，それを表現する時に適当に省略するか，忠実に守るかといったことがあります．

したがって与えられた図形をもとにして，3次元図形を描くにはさまざまなパラメータが必要になってくるのですが，ポイントになる計算だけをして，あとは省略してしまい，それらしく見えればよいという程度に考えることにしましょう．

＊＊＊3つの回転軸と位置の計算

まず立体物の回転を考えましょう．ビルなどは回転できませんが見る人がその位置を変えれば，相対的にビルを回転させるのと同じになります．ここでは見る人は動かず，対象になるものを回転させることにしておきます．空間にある点は，X，Y，Zと3つの位置の情報が与えられており，回転をする軸もおなじようにX軸，Y軸，Z軸になります．このうち，もっとも分りやすいのはZ軸を中心に回転させた

▲3次元図形と3つの回転軸

▲Z軸を中心にした回転は平面図形の回転とおなじ

時で，これは平面形の回転とおなじことになります．

すなわち，点の位置情報のうち奥行きに関係するもの（点Z）はいっさいその値を変えずに，点XがX′，点YがY′になります．

Z軸の回転：X→X′　Y→Y′　Z→Z

同様にしてY軸を中心にして回転させる（たとえば玄関のドアを開けるような回転）と

Y軸の回転：X→X′　Y→Y　Z→Z′

X軸を中心にして回転させる（たとえば洗濯機のフタを開けるような回転）と

X軸の回転：X→X　Y→Y′　Z→Z′

というように変化します．これらはいずれも平面図形の回転の計算式がそのまま使えることになります．

● Z軸を中心にした回転

$x' = x\cos\theta_z - y\sin\theta_z$

$y' = x\sin\theta_z + y\cos\theta_z$

● Y軸を中心にした回転

$z' = z\cos\theta_y - x\sin\theta_y$

$x' = z\sin\theta_y + x\cos\theta_y$

● X軸を中心にした回転

$y' = y\cos\theta_x - z\sin\theta_x$

$z' = y\sin\theta_x + z\cos\theta_x$

というわけです．

奥行きの情報を使って平面図にする

＊ディスプレイは平面なので……

ビルディングなど，稜線が平行な建物などを見ると分るのですが，もともとは平行な線は，遠くに行くほど間隔がせまくなり，やがてはどこかで交わるように見えます．この，仮想的に交わる点を消点といいますが，立体物を平面上に描くときは，この消点をいくつに設定するかによってその技法は異なります．

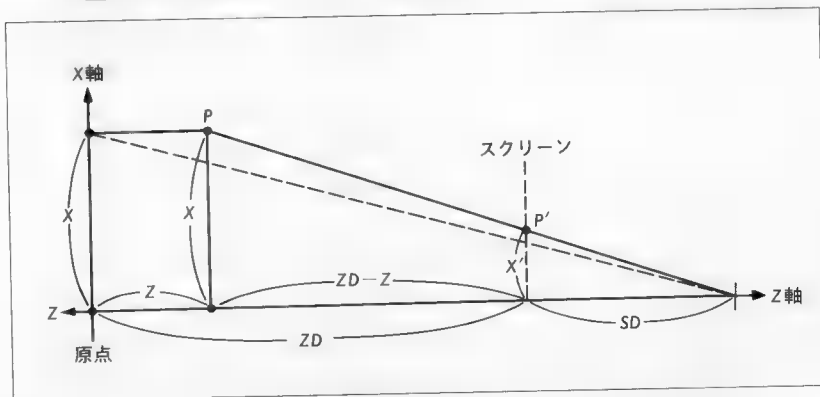
ここでは単（1点）消点による透視を考え，それをプログラム化することを考えましょう．単点透視は，たとえば立体物の上下と左右の線は平行（もともとの立体物の稜線が平行として）に描かれ，奥行きに関係する線が平行にならず，消点に向って収束するような図になる，もっともやさしいものですがこれで十分に通用します．

単点透視の基本は

1 立体物の奥行きに関係するデータを知り

2 奥行きのデータを使い，平面図の点（X, Y）を変換する

という，2つの作業を行ないます．この変換の例を下の図で説明しましょう．

左の四角が実際の物で，これをスクリーンに写して，X座標がいくつになるのかを計算しよう，というわけです．いま，Z軸の正の方向（見ている人にとっては手前側）のZの位置に，原点からXだけ離れた点Pがあるとします．なお，ここでは点PのYの値については考えには入れません．このスリーンからSDだけ離れた所に，とりあえず人が立っているとしましょう．実際の点Pとスクリーンの点P′およびS

DとZDは，比例関係になっていることが一目で分ります．

P：P′＝[(SD)＋(ZD－Z)]：SD

ですから，この関係からP′のX座標値を求めることができます．

P×SD＝P′×[(SD)＋(ZD－Z)]

P′＝P×SD÷[(SD)＋(ZD－Z)]

図の点線はスクリーンに写した奥にある点の場合で，スクリーン上には手前の点Pよりも下に投影されることが分ります．もともとの立体物は，点Pも，その点よりも奥にある点も，X座標については同じであったのですが，スクリーンに写した時はX座標値に関しては小さな値になることが分ります．

これが変換の作業で，スクリーンをディスプレイと考えれば，本物の立体物の奥行きに関係する要素（Z座標値）は，変換によって消えてしまい，平面に描くことができるのです．なお，Y座標についてもまったく同じ計算式が使えます．

＊＊変換についての注意点は

また，この図を見て分ることですが，スクリーンを左に近づければ平面に写される図は大きくなりますし，それと共にもともとの立体物の寸法は変らないのに，平面図にした時に奥行きについての表現が強張されます．また，SDを大きくとると写される図の大きさは実物に近くなり，奥行きについての表現は弱まり，すべての線が平行に近くなります．

ZDやSDの値をどう設定するかについての法則はないので，実際のプログラムではそれらしく見える値にしてよいでしょう．また，ZDは省略して，変換後のX座標を求めるのに

X′＝X×SD÷(SD－Z)

としてもかまいません．

プログラミングに際しては，立体物の点の情報(X, Y, Z)をしっかりとつかまえていれば，この立体物を傾けたりした状態を平面に写しとることもたやすくできます．また，点の情報を多次元の配列に収めれば，プログラムは短かい行数になります．たとえば立体物の点が10個だったらP(10, 10, 10) などとし，最初の添字が点の数，つぎがX座標，最後がY座標といったように使えます．ただし，慣れないうちは1次元の配列を使い，変数名も分りやすくする（例：手前の点のX座標はTX，奥はOXなど）ほうがまちがいありません．

立体の回転プログラム

＊直方体をサンプルに

では，立体物をX，Y，Z軸を中心にして回転させ，それを平面図上に表わすプログラムを作りましょう．ここではサブルーチンを多くして，個々の操作がよく分るようにしています．これを元にして応用するときはそれをふまえて要領よくまとめれば半分ほどの行数に圧縮できると思います．

ここでモデル化した立体物は幅300，高さ200，奥行き150，3次元軸に対しては上下と左右については中央の位置に置いています．変数TX，TY，TZについてはZ軸に対して正の方向，OX，OY，OZは負の方向になっています．したがって正面から見たとき，OXやOYのほうが小さく見えます．まず原図を出すために行番号170～250で，正立し正面から見たものを出力します．なお，ここではSD＝1000，ZD＝400に設定しました．

＊＊これをもとに，いろいろと応用してみよう

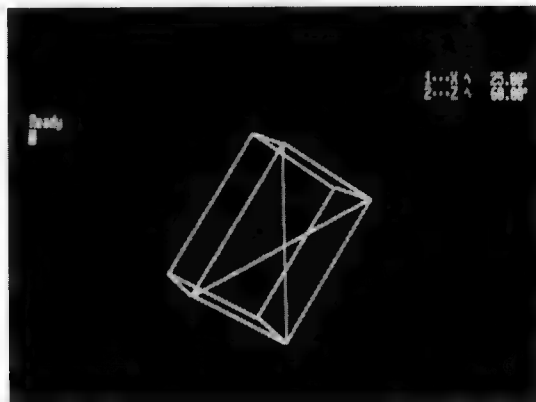
立体物はそれぞれの軸について，何回でも回転させることができます．回転をさせたら，当然のことですが点のX，Y，Zの座標値は変化します．ここではTX(I)，TY(I)，TZ(I)などの，回転をさせる前の配列を使うと共に，1回の回転ごとにそれをRTX(I)，RTY(I)などという新しい配列に一度変換してやります．すべての座標計算が終ったら，この座標値をまたTX(I)，TY(I)にもどしておくことにより，それぞれの軸を中心にした回転サブルーチンを何回も呼び出すことができるのです(行番号630，730，830)．なお，それぞれの回転を記録しておくために，回転軸と回転角を配列に入れました（行番号350）回転を何回命令したかは，Rという変数でカウントしています．

また，この立体物の，最初は手前に見える面には×の対角線を引いているので，回転をさせたあとにその面がどこに位置しているかが分ります（行番号1100～1110および写真参照)．なお，回転をさせた時の座標計算用に，たとえばDEF　FNを使い，回転をさせる前に変数を引渡しておく（例：X軸の角度はDEGXで入力し，計算ルーチンの前にDEG＝DEGXなどとする)などの手法を使えば，ひとつのサブルーチンですみます．

サンプルの図形にこだわらず，自由な形のもので試して下さい．

```
100 '************************ ﾘｯﾀｲﾉ ｶｲﾃﾝ ****************************
110 '
120   WIDTH80,25:PAI=3.14159:SY=.425
130   X0=300:Y0=100:RAD=PAI/180:SD=1000:ZD=400
140 GOSUB 1140:'ｻﾞﾋｮｳ DATAｦ ﾖﾑ
150 '
160 '---------------- ｹﾞﾝｽﾞ ﾋｮｳｼﾞ ------------------
170   C=7:CLS
180   CONNECT(50,100)-(550,100),6:'....................Xｼﾞｸ
190   CONNECT(300,15)-(300,180),6:'....................Yｼﾞｸ
200   SYMBOL(40,97),"X",1,1,7
210   SYMBOL(296,5),"Y",1,1,7
220 DEGZ=0*RAD:'....................................ｻｲｼｮﾊ ｶｲﾃﾝｼﾅｲ
230 GOSUB 760:'.................................Zｼﾞｸ ROTATE
240 GOSUB 860:'.................................XY ﾍﾝｶﾝ
250 GOSUB 950:'.................................ｽﾞｹｲｦ ｶｸ
260 '
270 '---------------- ｶｲﾃﾝﾋｮｳｼﾞ --------------------
280 R=1:C=5:RESTORE:GOSUB 1140:'.....................DATAｦ ﾖﾐﾅｵｽ
290   LOCATE 0,0
300   PRINT"X ｼﾞｸ ﾏﾜﾘ --->1"
310   PRINT"Y ｼﾞｸ ﾏﾜﾘ --->2"
320   PRINT"Z ｼﾞｸ ﾏﾜﾘ --->3"
330   LOCATE 0,4:INPUT"ﾄﾞﾚﾆ ｼﾏｽｶ ";MENU
340   ON MENU GOSUB 420,460,500
350 ROTATE$(R)=R$:DEG(R)=DEG:'.......................ｶｲﾃﾝ ｷﾛｸﾖｳ
360 LOCATE 0,5:PRINT"ﾋｮｳｼﾞ･･･>H ｶｲﾃﾝ･･･SPACE BAR";
370   WHICH$=INKEY$:IF WHICH$="" THEN 370
380   IF WHICH$="H" OR WHICH$="h" THEN CLS:GOSUB 870:GOSUB 950:GOSUB 1380:END
390   IF ASC(WHICH$)=&H20 THEN GOSUB 1360:R=R+1:GOTO 330
400 GOTO 360
410 '
420 '------------- Xｼﾞｸ ｶｲﾃﾝ ﾆｭｳﾘｮｸ --------------
430 INPUT"ｶｸﾄﾞﾊ  ";DEGX:DEG=DEGX:R$="X":DEGX=DEGX*RAD
440   GOSUB 550:'.........................ｶｲﾃﾝ ｹｲｻﾝ
450 RETURN
460 '------------- Yｼﾞｸ ｶｲﾃﾝ ﾆｭｳﾘｮｸ --------------
470 INPUT"ｶｸﾄﾞﾊ  ";DEGY:DEG=DEGY:R$="Y":DEGY=DEGY*RAD
480   GOSUB 650:'.........................ｶｲﾃﾝ ｹｲｻﾝ
490 RETURN
500 '------------- Zｼﾞｸ ｶｲﾃﾝ ﾆｭｳﾘｮｸ --------------
510 INPUT"ｶｸﾄﾞﾊ  ";DEGZ:DEG=DEGZ:R$="Z":DEGZ=DEGZ*RAD
520   GOSUB 750:'.........................ｶｲﾃﾝ ｹｲｻﾝ
530 RETURN
540 '
550 '------------- Xｼﾞｸ ｶｲﾃﾝ ｹｲｻﾝ ----------------
560  FOR I=0 TO 3
```

```
570    RTY(I)=TY(I)*COS(DEGX)-TZ(I)*SIN(DEGX)
580      RTZ(I)=TY(I)*SIN(DEGX)+TZ(I)*COS(DEGX)
590      ROY(I)=OY(I)*COS(DEGX)-OZ(I)*SIN(DEGX)
600    ROZ(I)=OY(I)*SIN(DEGX)+OZ(I)*COS(DEGX)
610  NEXT I
620 'ｶｲﾃﾝ ｻﾞﾋｮｳｦ ﾓﾄﾉﾊｲﾚﾂﾆ ﾓﾄﾞｽ
630  FOR I=0 TO 3:TY(I)=RTY(I):TZ(I)=RTZ(I):OY(I)=ROY(I):OZ(I)=ROZ(I):NEXT
640 RETURN
650 '------------ Yｼﾞｸ ｶｲﾃﾝ ｹｲｻﾝ -----------------
660  FOR I=0 TO 3
670    RTX(I)=TX(I)*COS(DEGY)+TZ(I)*SIN(DEGY)
680      RTZ(I)=TZ(I)*COS(DEGY)-TX(I)*SIN(DEGY)
690      ROX(I)=OX(I)*COS(DEGY)+OZ(I)*SIN(DEGY)
700    ROZ(I)=OZ(I)*COS(DEGY)-OX(I)*SIN(DEGY)
710  NEXT I
720 'ｶｲﾃﾝ ｻﾞﾋｮｳｦ ﾓﾄﾉﾊｲﾚﾂﾆ ﾓﾄﾞｽ
730  FOR I=0 TO 3:TX(I)=RTX(I):TZ(I)=RTZ(I):OX(I)=ROX(I):OZ(I)=ROZ(I):NEXT
740 RETURN
750 '------------- Zｼﾞｸ ｶｲﾃﾝ ｹｲｻﾝ -----------------
760  FOR I=0 TO 3
770    RTX(I)=TX(I)*COS(DEGZ)-TY(I)*SIN(DEGZ)
780      RTY(I)=TX(I)*SIN(DEGZ)+TY(I)*COS(DEGZ)
790      ROX(I)=OX(I)*COS(DEGZ)-OY(I)*SIN(DEGZ)
800    ROY(I)=OX(I)*SIN(DEGZ)+OY(I)*COS(DEGZ)
810  NEXT I
820 'ｶｲﾃﾝ ｻﾞﾋｮｳｦ ﾓﾄﾉﾊｲﾚﾂﾆ ﾓﾄﾞｽ
830 FOR I=0 TO 3:TX(I)=RTX(I):TY(I)=RTY(I):OX(I)=ROX(I):OY(I)=ROY(I):NEXT
840 RETURN
850 '
860 '----------------- XY ﾍﾝｶﾝ  ---------------------
870   FOR I=0 TO 3
880     TX(I)=TX(I)*(SD)/(SD+ZD-TZ(I))
890       TY(I)=TY(I)*(SD)/(SD+ZD-TZ(I))
900       OX(I)=OX(I)*(SD)/(SD+ZD-OZ(I))
910     OY(I)=OY(I)*(SD)/(SD+ZD-OZ(I))
920   NEXT I
930 RETURN
940 '
950 '------------------ ｽﾞｹｲｦｶｸ ---------------------
960 FOR I=0 TO 3
970  TX(I)=TX(I)+X0:TY(I)=Y0-SY*TY(I)
980  OX(I)=OX(I)+X0:OY(I)=Y0-SY*OY(I)
990 NEXT I
1000 ' ﾀﾞﾐｰﾉ ｻﾞﾋｮｳｦ ﾂｸﾙ
1010 TX(4)=TX(0):TY(4)=TY(0)
1020 OX(4)=OX(0):OY(4)=OY(0)
1030 'ｽﾞｹｲｦ ｶｸ
```

```
1040  FOR I=0 TO 3
1050     CONNECT(TX(I),TY(I))-(TX(I+1),TY(I+1)),C
1060       CONNECT(OX(I),OY(I))-(OX(I+1),OY(I+1)),C
1070     CONNECT(OX(I),OY(I))-(TX(I),TY(I)),C
1080  NEXT I
1090 'ﾀｽｷｶﾞｹﾉ ﾋｮｳｼﾞ
1100    CONNECT(TX(0),TY(0))-(TX(2),TY(2)),C
1110    CONNECT(TX(3),TY(3))-(TX(1),TY(1)),C
1120 RETURN
1130 '
1140 '--------------- DATAｦ ﾖﾑ -------------------
1150   FOR I=0 TO 3
1160      READ TX(I),TY(I),TZ(I):'・・・・・・・・・・・ﾃﾏｴﾉ ﾃﾝ
1170   NEXT I
1180 '
1190   FOR I=0 TO 3
1200      READ OX(I),OY(I),OZ(I):'・・・・・・・・・・・ｵｸﾉ ﾃﾝ
1210   NEXT I
1220 RETURN
1230 '
1240 'ｽﾞｹｲ DATA <<ﾃﾏｴ>>
1250 DATA -150,-100 ,50:'TX,TY,TZ  0
1260 DATA  150,-100 ,50:'TX,TY,TZ  1
1270 DATA  150, 100, 50:'TX,TY,TZ  2
1280 DATA -150, 100, 50:'TX,TY,TZ  3
1290 'ｽﾞｹｲ DATA <<(ｵｸ)>>
1300 DATA -150,-100,-100:'OX,OY,OZ 0
1310 DATA  150,-100,-100:'OX,OY,OZ 1
1320 DATA  150, 100,-100:'OX,OY,OZ 2
1330 DATA -150, 100,-100:'OX,OY,OZ 3
1340 '
1350 'ﾒｯｾｰｼﾞｹｼ
1360 LOCATE 0,5:PRINT SPACE$(29);:RETURN
1370 'ｶｲﾃﾝｷﾛｸﾉ ﾋｮｳｼﾞ
1380 FOR I=1 TO R
1390   LOCATE 60,I
1400   PRINT USING"##・・・&&ﾍ ####.##°";
   I;ROTATE$(I);DEG(I)
1410 NEXT
1420 RETURN
```

グラフィックカーソルの基本

*いろいろあるカーソルの処理命令

カーソル（cursor）は，なにかを読取るための指標といった意味で，テキスト画面についてはLOCATE, POS, CSRLINが使われ，グラフィック画面に対してはGCURSOR（graphic cursor）が使われます．GCURSORの基本は

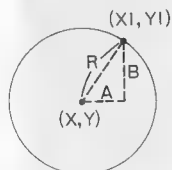
GCURSOR（出力位置），（読取り変数名）

という形になります．Gカーソルが画面に出力されたら，カーソル移動およびシフトキーを使い，目的の位置まで動かし，リターンキーを押すと，指定した変数名にその位置が代入されるもので，機能としてはライトペンに通じるものです．

**直線と四角は2点をそのまま利用する

サンプルプログラムはGカーソルを使ってまず適当なところに十文字のカーソルを出力し（行番号140），リターンキーが押されると，その位置に黄色の点を打ちます（行番号150）．つぎに画面に線を引くのか，四角を作るのか，円を描くのかを求め（行番号170～190），つぎにさきほど打った点から少し離れた位置にGカーソルを出し，もうひとつの点の入力を待ちます（行番号210）．リターンキーが押されたら，それぞれの指定ルーチンを呼出し，グラフィックスを実行して終ります．

LINE文は指定の2点間を結ぶものですし，四角も同様に指定した2点を対角とするものですから，このプログラムにおける2つの点(X, YおよびX1, Y1）をそのまま使ってよいのですが，円はこのままでは描けません．

***2点を使って円を描くには

R＝SQR (A^2+B^2)
▲2点を指定した円の考えかた

指定をした点のうち（X, Y）を中心とする円は，もうひとつの指定点（X1, Y1）が円周上にあるもの，という前提のプログラムですから，半径を計算しないといけないのです．半径はピタゴラスの定理によって

半径＝直角三角形の斜辺

ですから，X座標，Y座標の線分（それぞれ直角三角形の，直角をはさむ2つの辺の長さ）をもとめ，計算によって求めることになります．行番号330が半径算出の文で，それぞれの座標の x, y 線分から半径を求めています．なお，画面のタテヨコ比に合せた係数を使い，補正をし

```
100 '****************** GCURSOR 1   ｽﾞｹｲｦ ｶｸ **********************
110 SY=.43:WIDTH 80,25:CLS
120 '
130 LOCATE 0,0:GOSUB 280 '...................ﾒｯｾｰｼﾞ SUB
140 GCURSOR(300,100),(X,Y) '.................RETURNﾃﾞ X,Y ｦ ﾓﾄﾒﾙ
150 PSET(X,Y,6) '............................ｷｲﾛﾃﾞ PSET
160 '
170 LOCATE 0,0:PRINT"1:LINE   2:BOX   3:CIRCLE ﾄﾞﾚﾆ ｼﾏｽｶ";
180 MENU$=INKEY$:IF MENU$="" THEN 180
190 MENU=VAL(MENU$):IF MENU<1 OR MENU>3 THEN 180
200 GOSUB 280 '..............................ﾒｯｾｰｼﾞ SUB
210 GCURSOR(X+5,Y+5),(X1,Y1) '...............RETURNﾃﾞ X1,Y1 ｦ ﾓﾄﾒﾙ
220 GOSUB 290 '..............................ﾒｯｾｰｼﾞｹｼ SUB
230 ON MENU GOSUB 310,320,330
240 LOCATE 0,24:PRINT USING"X=###:Y=###    X1=###:Y1=###";X;Y;X1;Y1;
250 '
260 FOR WAIT=0 TO 6000:NEXT:CLS:GOTO 130
270 '
280 LOCATE 45,0:PRINT"GCURSORｦ ｳｺﾞｶｼﾃ ｽｷﾅﾄｺﾛﾃﾞ RETURN":RETURN
290 LOCATE 0,0:PRINT STRING$(78," "):RETURN
300 '
310 LINE(X,Y)-(X1,Y1),PSET,6:RETURN
320 LINE(X,Y)-(X1,Y1),PSET,6,B:RETURN
330 R=SQR(ABS(X-X1)^2+ABS((Y-Y1)/SY)^2) '................ﾊﾝｹｲﾉ ｹｲｻﾝ
340 CIRCLE(X,Y),R,6,SY:PRESET(X,Y):RETURN
```

た円を描かせないといけません.

****その他の注意点

Gカーソルは一度に最大10点までの座標を読みとることができ，たとえばすでに出力されているグラフィック画面の点を，つぎつぎと読みとることもできます．逆に，なにか自由に描きたい図形があった時も，ためしにGカーソルを出力してその座標値を読みながら，およその形を見ることもできるわけです．

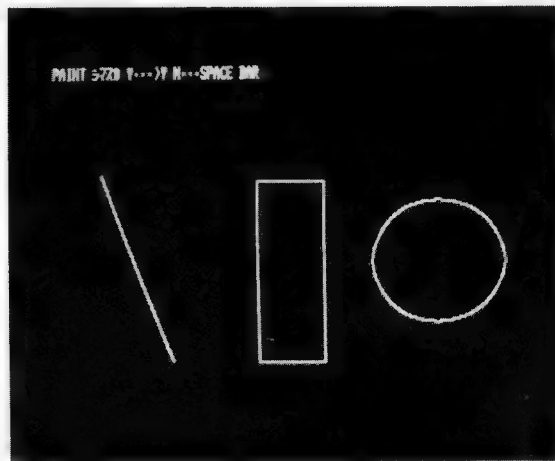
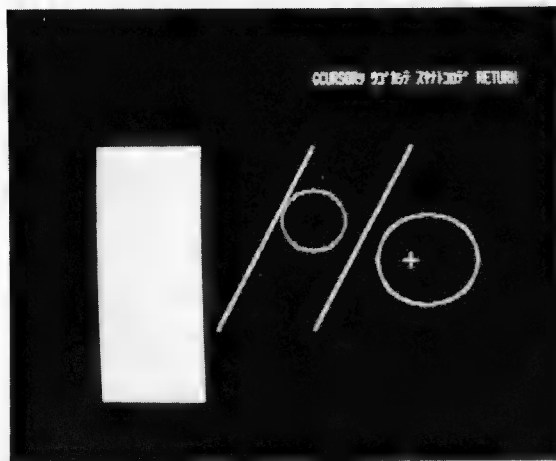
グラフィックカーソルで絵を描く

＊終点を始点にしてつぎつぎと描く

グラフィックカーソルを使って，画面上につぎつぎと絵を描いてゆくプログラムを作りましょう．前のページに出てきた考えかたを基本にして，連続して絵を描くと共に，ペイントもできるようにしています．まず行番号110で描き始めの点XS，YSをきめて，Gカーソルを動かし，最初にリターンした点にPSETします（行番号150）．

線を引くのか，四角か，円にするのかを求め，つぎに何色にするかを求め（行番号220～230），もうひとつの点を入力したら，さいしょの点は消してしまいます（行番号250）．この処理を終えたら，指定の図形を描くサブルーチンを呼び，実行します．

つぎはペイント作業になりますが，線を引くモードを選んだ時はペイントしませんので，ペイントルーチンをパスして，新しい入力を求めることにしています（行番号270）．ペイントのためには，どの位置からペイントさせるかを命令する必要があるため，もういちどGカーソルを画面に出力し，ペイント点XP，YPを求めています（行番号350）．なお，このプログラムでは四角と円についてペイントしますが，できあがってゆく絵を自由にペイントするためには，たとえばペイントルーチンを単独に設定してもいいでしょう．

画面上につぎつぎと絵を描くために，ひとつの絵を描いたら，終点（XP，YPまたはX1，Y1）を始点に変えています（行番号390，410）．

```
100 '******************* GCURSOR 2   ｽﾞｹｲｦ ﾂﾂﾞｹﾃ ｶｸ *******************
110 SY=.43:XS=300:YS=100:WIDTH 80,25:CLS
120 '
130 LOCATE 0,0:GOSUB 450 '･･･････････････････ﾒｯｾｰｼﾞ SUB
140 GCURSOR(XS,YS),(X,Y) '･･･････････････････RETURNﾃﾞ X,Y ｦ ﾓﾄﾒﾙ
150 PSET(X,Y,6) '･････････････････････････････ｷｲﾛﾃﾞ PSET
160 '---------------------- ﾓｰﾄﾞ ｾﾝﾀｸ ---------------------------------
170 LOCATE 0,0:PRINT"1:LINE   2:BOX   3:CIRCLE ﾄﾞﾚﾆ ｼﾏｽｶ";
180 MENU$=INKEY$:IF MENU$="" THEN 180
190 MENU=VAL(MENU$):IF MENU<1 OR MENU>3 THEN 180
200 GOSUB 450 '·······························ﾒｯｾｰｼﾞ SUB
210 '---------------------- ｲﾛﾉ ｾﾝﾀｸ -----------------------------------
220 LOCATE 0,1:INPUT"ｲﾛﾊ ﾅﾆｲﾛ  ";C$:C=VAL(C$)
230 IF C<0 OR C>7 THEN 220
240 GCURSOR(X,Y),(X1,Y1) '･･･････････････････RETURNﾃﾞ X1,Y1 ｦ ﾓﾄﾒﾙ
250 PRESET(X,Y) '･････････････････････････････ｻｲｼｮﾉ ｲﾛｦ ｹｽ
260 ON MENU GOSUB 490,510,530 '･･････････････ｼﾃｲ ｽﾞｹｲｦ ｶｸ
270 IF MENU=1 THEN 410 '･･････････････････････LINE ﾅﾉﾃﾞ PAINT ﾊ ﾊﾟｽ
280 GOSUB 460 '·······························ﾒｯｾｰｼﾞｹｼ SUB
290 '--------------------- PAINT ﾆｭｳﾘｮｸ ------------------------------
300 LOCATE 0,0:PRINT"PAINT ｼﾏｽｶ Y･･･>Y N･･･SPACE BAR"
310 P$=INKEY$:IF P$="" THEN 310
320 IF ASC(P$)=&H20 THEN 410 ELSE IF P$="Y" OR P$="y" THEN 330 ELSE 310
330 LOCATE 0,1:INPUT"ｲﾛﾊ ﾅﾆｲﾛ";CP$:CP=VAL(CP$)
340 IF CP<0 OR CP>7 THEN 330
350 GCURSOR(X1,Y1),(XP,YP)
360 GOSUB 450 '·······························ﾒｯｾｰｼﾞ SUB
370 PAINT(XP,YP),CP,C '･･･････････････････････PAINT
380 '
390 XS=XP:YS=YP:GOTO 420 '････････････････････GCURSORｦ ﾀﾞｽｲﾁｦ ｷﾒﾙ
400 '
410 XS=X1:YS=Y1'･･････････････････････････････GCURSORｦ ﾀﾞｽｲﾁｦ ｷﾒﾙ
420 GOSUB 460:GOTO 130
430 '
440 '========================== ｻﾌﾞﾙｰﾁﾝ ===============================
450 LOCATE 45,0:PRINT"GCURSORｦ ｳｺﾞｶｼﾃ ｽｷﾅﾄｺﾛﾃﾞ RETURN":RETURN
460 LOCATE 0,0:PRINT STRING$(159," "):RETURN
470 '
480 ' LINE
490 LINE(X,Y)-(X1,Y1),PSET,C:RETURN
500 ' BOX
510 LINE(X,Y)-(X1,Y1),PSET,C,B:RETURN
520 ' CIRCLE
530 R=SQR(ABS(X-X1)^2+ABS((Y-Y1)/SY)^2) '････････････････ﾊﾝｹｲﾉ ｹｲｻﾝ
540 CIRCLE(X,Y),R,C,SY:RETURN
```

カラーパレットの使いかた

＊カラーパレットとカラーコード

パレットに絵具をおいてゆく

V 3.0以降のF-BASICでは，色の処理についてパレットという考え方を導入しています．パレット（palette）は絵具をしぼり出してのせておく板のことですが，絵具を入れる場所は8つあり，それぞれ0～7までの番号がつけられています．このうち0番は黒しか受けつけないので，実質は1～7番までのパレット番号（コード）に，好きな色を指定することができます．

さて，パレットとは別にカラーコードがあり，FM-77で使える8色は以下のように定義されています．

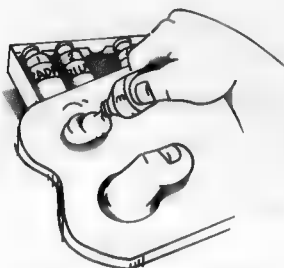
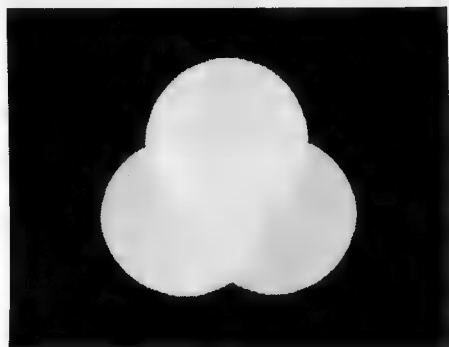
0……黒　1……青　2……赤　3……シアン（紫）

4……緑　5……水色　6……黄　7……白

光の3原色はR・G・Bで，カラーコードの順からいうとB・R・G…つまり1・2・4の組合せになります．この3つさえおぼえていれば，1＋2＝(青＋赤)＝紫(3)，1＋4＝(青＋緑)＝水色(5)など，原色を合成した色が何色でどのコードかがすぐに分ります．

サンプルプログラムはB・R・Gの3つの画面をまず単独で操作したのち，同時に出力するもので，いま話した色の関係がよく分ります．

＊＊色をつぎつぎと変える

カラーパレットの操作は，パレットコードにカラーコードを当てはめることですから，基本的には単純なものです．さきほどのプログラムの行番号220を

▲右のプログラムの実行例

```
100 '******* 3ｹﾞﾝｼｮｸ ﾄ ｺﾞｳｾｲ ******
110 CLS
120 SCREEN 1,1:'······BLUE  ｶﾞﾒﾝ ON
130 CIRCLE(220,130),140,,,,,F
140 '
150 SCREEN 2,2:'······RED   ｶﾞﾒﾝ ON
160 CIRCLE(380,130),140,,,,,F
170 '
180 SCREEN 4,4:'······GREEN ｶﾞﾒﾝ ON
190 CIRCLE(300,70),140,,,,,F
200 '
210 SCREEN 7,7:'3ｶﾞﾒﾝｦ ﾄﾞｳｼﾞﾆ ON
220 GOTO 220
```

220 FOR I=1 TO 7：COLOR=(I,7)：NEXT

に変えると，３つの円はすべて白い色になります．すべてのパレットコードに白い色の７を指定したからです．元に戻すには

FOR I=1 TO 7：COLOR=(I,I)：NEXT

とします．下のサンプルはパレットに色をあてはめてゆく２つの例を示しています．実際に試してその効果を見てください．

```
100 '======================= ｶﾗｰﾊﾟﾚｯﾄﾉ ｿｳｻ ==============================
110 X0=300:Y0=100:SY=.43
120 GOSUB 480:'･･･････････････････････････････････ﾊﾟﾚｯﾄｦ ｼｮｷﾁ ﾆ ﾓﾄﾞｽ SUB
130 '**** ｷﾎﾝ ｽﾞｹｲｦ ｴｶﾞｸ
140 CLS
150 C=1
160 FOR PAI=0 TO 2*3.14 STEP 2*3.14/7
170   X=X0+120*COS(PAI):Y=Y0-120*SY*SIN(PAI):'･････ｴﾝｼｭｴｼﾞｮｳﾆ ｴﾝｦ ﾊｲﾁ
180      CIRCLE (X,Y),40,C,,,,F:'･････････････････ﾇﾘﾂﾌﾞｼﾉ ｴﾝｦ ｴｶﾞｸ
190   C=C+1:IF C=8 THEN 220:'･････････････････････7ﾂﾉ ｴﾝｦ ｶｲﾀﾗ ﾀﾞｯｼｭﾂ
200 NEXT PAI
210 '
220 FOR I=1 TO 7:COLOR=(I,7):NEXT:'････････････････ｽﾍﾞﾃﾉｲﾛｦ ｼﾛﾆｽﾙ
230 WHILE KAISU<50
240 '**** ﾊﾟﾚｯﾄ1
250   FOR I=1 TO 7:IF I=7 THEN C=0 ELSE C=I
260     COLOR=(I,C):'･････････････････････････････ｼﾞｭﾝﾊﾞﾝﾆ 7ﾂﾉｲﾛﾆｽﾙ
270     FOR T=0 TO 80:NEXT T:'････････････････････ｼﾞｶﾝﾁｮｳｾｲ
280   NEXT I
290 '
300   FOR I=1 TO 7:COLOR=(I,7):NEXT:'･････････････ｽﾍﾞﾃﾉｲﾛｦ ｼﾛﾆｽﾙ
310 KAISU=KAISU+1
320 WEND
330 '
340 GOSUB 480:'･･･････････････････････････････････ﾊﾟﾚｯﾄｦ ｼｮｷﾁ ﾆ ﾓﾄﾞｽ SUB
350 '**** ﾊﾟﾚｯﾄ2
360 C=1
370 WHILE KAISU<100
380    FOR I=1 TO 7
390      C=(C MOD 7)+1:'･･･････････････････････････ﾊﾟﾚｯﾄｿｳｻ
400         FOR T=0 TO 20:NEXT T:'････････････････ｼﾞｶﾝﾁｮｳｾｲ
420      COLOR=(I,C):'･････････････････････････････ﾊﾟﾚｯﾄﾍﾉ ﾜﾘｱﾃ
430    NEXT I
440   C=C+1:KAISU=KAISU+1
450 WEND
460 END
470 '
480 FOR I=1 TO 7:COLOR=(I,I):NEXT
490 RETURN
```

VRAMとF-BASICにおける操作

＊VRAMとは

FM-77にはディスプレイに出力するための映像情報を扱う専用のメモリが，メインメモリとは別に用意されています．VRAM（ブイラム：Video用Random Access Memory）と呼ばれるもので，合計で48Kバイト分になります．

ディスプレイのドット（点）は，横が640個，縦は200個あるので，合計では128,000個になります．ただしこの数はディスプレイがモノクロ（単色）の時で，FM-77のように光の3原色（B・R・G）に対応しているものでは，B用にも，R用にも，G用にも128,000個分のドットを点灯するためのメモリが必要になり，けっきょく128,000×3＝384,000個分のメモリが必要になります．

ひとつのドットを点灯させるために1ビットが使われるので，ビデオ用RAMは48,000バイト（384,000÷8）になるのです．

＊＊3種類のVRAMをコントロール

これらのメモリが，たとえばR用のメモリがディスプレイのR入力端子に直結されているとしたら，BASICによる操作は赤い点を光らせるか，光らせないかだけになります．ところが実際のF-BASIC（V3.0）では，これらのR・B・G用のメモリとディスプレイ出力とを直結させず，いろいろな形でコントロールしています．

カラーパレットもその例で，パレット番号とカラーコードの組合せも，実はVRAMを操作しているのです．

```
COLOR=(5, 1)
```

という文は，COLOR 5という命令が与えられると，B（1が割当てられている）のVRAMからの映像用情報を取出して，ディスプレイに出力することを意味します．同様に

```
SCREEN 3, 5
```

はB(1)用とR(2)用のVRAMに書込みをして，ディスプレイへの出力はB(1)用とG(4)用のVRAMから，と設定します．

そのほか，LINE文やCIRCLE文その他のグラフィック命令における論理演算機能も同じで，指定した色と，画面に出力されているドットのパレット番号を演算して，その結果の色でディスプレイをするのです．

5

グラフィックスの進んだ使いかた

GET@と配列の大きさ

＊GET@の使いかた

GET@はVRAMに書込まれている情報を取出して，プログラム用のメモリに移す作業です．ふつうのグラフィックスでは，たとえば円を描いたり，ペイントするのにある程度の時間がかかるのに対し，GET@したものをPUT@すると，瞬時にグラフィックスが再現できる利点があります．したがってGET@，PUT@の使い方としてはつぎのようなことが考えられます．

○同じパターンを，多くの場所に出力する時
○アニメーションのように，動く絵を出力する時
○グラフィックスをディスケットに収める時
○トランプやマージャンゲームその他，各種のパターンを出力する時

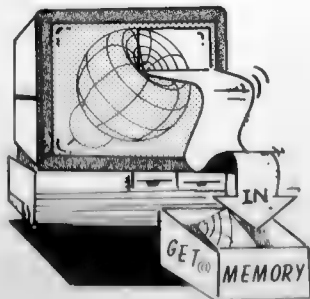
＊＊GET@と4つの形式

GET@，PUT@には4つの形式があり，目的によって使い分けますが，まずその形式を見ましょう．GET@は

形式1　キャラクタをメモリに移す(色はテキスト画面と同じ)
　　GET@(座標)，配列名
形式2　キャラクタと色の情報(パレット情報)をメモリに移す
　　GET@A(座標)，配列名
形式3　ドットパターンと選択する色を指定してメモリに移す
　　GET@(座標)，配列名，G，パレットコード，…
形式4　ドットパターンをフルカラーでメモリに移す
　　GET@A(座標)，配列名，G

この形式を見て気がつくことは，アルファベットのAとGの使い方です．

A：オール(all)といった意味で，パレットコードの情報もいっしょにメモリに入れてしまう．
G：グラフィックスといった意味で，ドットの情報（Gがつかないものはキャラクタコード情報）をメモリに入れる．

なお，これらの4つの使いわけにはルールはありません．ただし用意するメモリ（配列）は形式4がいちばん多くなります．メモリが多くてもよいのなら，形式4を使って形式1や形式2の代用…キャラクタを配列にとりこむこともできるのです．なお，配列はすべて整数型

でよいので，GET@のときは常に整数型を使うようにします．

***必要な配列の大きさについて

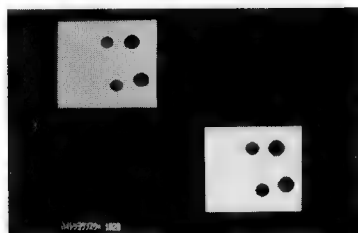
GET@するときには配列をあらかじめ用意しなければいけません．たとえば1キャラクタをGET@するのに「DIM GC(500)」としてもよいのですが，これでは用意する配列が大きすぎて，ムダになります．およその目印としては，整数型では

形式1　GET@する文字数に1を加えて，2で割る
形式2　形式1の2倍
形式3　GET@する面積を16で割り，8～20を加える
形式4　形式3の3倍

になります(正確な計算法はマニュアルを参照)．

下のプログラムは四角を描き，その中に円がいくつかあって，中がくりぬかれている図形をGET@，PUT@する例です．ここでは形式3の，パレットコード指定形のGET@を使いました．

GET@する範囲は(0,0)から(X, Y)とします．すると用意すべきGET@用の配列は行番号120になります．ここでは最後に20を加えていますが，8，16，20などと，いくつかの異なる数値を加えて，試してください．なお配列要素が小さい時は，一般の配列時のエラーメッセージ(Subscript Out Of Range)ではなく，イリーガル・ファンクションコールになるので注意して下さい．

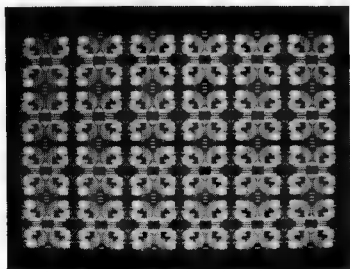
行番号200でGET@するのはパレットコード1（ここでは青)だけを指定しているので，行番号160，170の，5番のパレットコードによる図形（ここでは円）は配列には取込まれません．出力は行番号210で，3番のパレットコード（紫）を使っています．

▲右のプログラムの実行例

```
100 '============== GET@ * ｹｲｼｷ 3 * ==================
110 X=200:Y=80:'····················ﾒﾝｾｷ ｹｲｻﾝ&ｻﾞﾋｮｳﾖｳ
120 H=INT((X*Y)/16)+20:DIM G1%(H):'·····H=ﾊｲﾚﾂﾖｳｿｽｳ
140 'ｹﾞﾝｽﾞｦ ｴｶﾞｸ
150 WIDTH 80,25:CLS
160   CIRCLE(100,20),15,5:CIRCLE(150,20),18,5
170   CIRCLE(120,60),15,5:CIRCLE(170,55),18,5
180   LINE(0,0)-(X,Y),PSET,1,B:PAINT(1,1),1,5
190 '
200 GET@(0,0)-(X,Y),G1%,G,1:'············ｹｲｼｷ3 ﾉ GET@
210 PUT@(300,100)-(300+X,100+Y),G1%,PSET,3:'····PUT@
230  LOCATE 0,24:PRINT"ﾊｲﾚﾂﾖｳｿｽｳ=";H;
240  GOTO 240
```

GET@とデータのSAVE

*ひとつの絵から異なるものを作るには

GET@，PUT@は，1対1の対応を正しく守ると元の絵がそのままメモリに取込まれますが，GET@/PUT@の順序を変えることで，元とは異なったパターンのものも作成可能です．その例として乱数を使って作った塗りつぶしの四角から，合計4つのパターンを作成し，画面いっぱいに展開するカレイドスコープ（万華鏡（まんげきょう））を考えましょう．ここでは軸対称（X軸，Y軸）と点対称の3つのパターンをプログラムの上で作ります．その原理はGET@する順序を変えるという，簡単なものです．FOR～NEXTの2重ループを使い，1点1点をGET@，PUT@し，それをひとまとめにしてGET@します．

**パターンデータをファイルに収める

1点1点の操作ですから完成までに少し時間がかかるので，RUNさせたあと，しばらくしてから画面にはきれいな模様が出ます．このパターンをディスケットに収めておいて，数秒間待つだけでカレイドスコープが再現できるようにしましょう．下のプログラムは再現用で，右のプログラムが作成用になっています．

ここではシーケンシャルファイルを使い，GET@用の配列要素数（変数DATNUM）と，図形（変数はLG）のデータを記録しています．このプログラムの場合，ドット数は約1,000個ですが，形式4を使うため，メモリをかなり用意していることもあって，4クラスタほどを使ってSAVEしています．GET@する面積が広くなればSAVE，LOADに時間がかかると共に，ディスケットも多くの部分を使うので，ほどほどのGET@を考える方がよいでしょう．

▲ディスプレイ例

作成したパターンを読出し，ディスプレイするプログラム▶

```
100 '****************** GET@ ﾃﾞｰﾀｦ ﾐﾙ ********************
120 DEFINT A-Z:INPUT"ﾌｧｲﾙﾒｲ";FL$
130 OPEN"I",#1,"0:GT@"+FL$:INPUT #1,DATNUM
150 DIM LG(DATNUM):FOR I=0 TO DATNUM:INPUT #1,LG(I):NEXT
160 CLOSE
170 '
180 CLS
190 FOR PY=0 TO 138 STEP 46:FOR PX=20 TO 520 STEP 100
200  PUT@A(PX,PY)-(PX+100,PY+46),LG,PSET
210 NEXT PX,PY
220 GOTO 220
```

```
100 '====================== GET@ PUT@ ﾄ ﾀｲｼｮｳ ｽﾞｹｲ ===========================
110 DEFINT A-Z:X=50:Y=23:H=INT(3*(X*Y)/16)+18:DIM SG(H),PG(3),LG(H*4+20)
120 REM SG=SMALL GET@:PG=POINT GET@:LG=LARGE GET@
130 GOSUB 490:'ｽﾞｹｲｦ ｴｶﾞｸSUB
140 '----------------------------- ｹﾞﾝｽﾞｹｲ -------------------------------------
150 GET@A(0,0)-(X,Y),SG,G:PUT@A(200,50)-(200+X,50+Y),SG,PSET
160 '----------------------- Yｼﾞｸﾆ ﾀｲｼｮｳﾅ ｽﾞｹｲ -------------------------------
170 FOR GY=0 TO Y:FOR GX=X TO 0 STEP -1
180       GET@A(GX,GY)-(GX,GY),PG,G
190       PUT@A(300-GX,50+GY)-(300-GX,50+GY),PG,PSET
200 NEXT GX,GY
210 '----------------------- Xｼﾞｸﾆ ﾀｲｼｮｳﾅ ｽﾞｹｲ ------------------------------
220 FOR GY=Y TO 0 STEP -1:FOR GX=0 TO X
230       GET@A(GX,GY)-(GX,GY),PG,G
240       PUT@A(200+GX,98-GY)-(200+GX,98-GY),PG,PSET
250 NEXT GX,GY
260 '------------------------- ﾃﾝﾀｲｼｮｳ ｽﾞｹｲ -----------------------------------
270 FOR GY=Y TO 0 STEP -1:FOR GX=X TO 0 STEP -1
280       GET@A(GX,GY)-(GX,GY),PG,G
290       PUT@A(300-GX,98-GY)-(300-GX,98-GY),PG,PSET
300 NEXT GX,GY
310 '************************ ﾋﾄｶﾀﾏﾘｦ GET@,PUT@ ******************************
320 GET@A(200,50)-(300,96),LG,G:CLS
340 FOR PY=0 TO 138 STEP 46:FOR PX=20 TO 520 STEP 100
350   PUT@A(PX,PY)-(PX+100,PY+46),LG,PSET
360 NEXT PX,PY
370 '------------------------- ﾃﾞｰﾀｦ ﾌｧｲﾙﾆ ｶｷｺﾑ ------------------------
380 LOCATE 0,24:PRINT"SAVEｼﾏｽｶ  Y or N  ";
390 YN$=INKEY$:IF YN$="" THEN 390
400 IF YN$="N" OR YN$="n" THEN LOCATE 0,24:PRINT STRING$(20," ");:GOTO 460
410 IF YN$="Y" OR YN$="y" THEN 420 ELSE 380
420 LOCATE 0,24:INPUT"ﾌｧｲﾙﾒｲ(5ﾓｼﾞ ｲﾅｲ)     ";FL$
430 OPEN"O",#1,"0:GT@"+FL$:DATNUM=H*4+20:'·····················ﾊｲﾚﾂﾖｳｿｽｳ
440 PRINT #1,DATNUM:'································ﾊｲﾚﾂﾖｳｿｦ ｶｷｺﾑ
450 FOR I=0 TO DATNUM:PRINT #1,LG(I):NEXT:CLOSE:LOCATE 0,24:PRINT"SAVE END
    ";
460 GOTO 460
470 '
480 '**  RANNDOM  ｽﾞｹｲ
490 CLS:IF TIME>30000 THEN TIME$="00:00:00"
500    RANDOMIZE TIME
510    FOR I=1 TO 100:X1=INT(RND*52):Y1=INT(RND*20):X2=INT(RND*20):Y2=INT(RND*9)
       :C=INT(RND*7+1):LINE(X1,Y1)-(X1+X2,Y1+Y2),PSET,C,BF:NEXT
520   CONNECT(14,1)-(35,1)-(48,7)-(48,16)-(35,22)-(14,22)-(2,16)-(2,7)-(14,1),0
530 RETURN
```

4つのキーでロケットを動かす

＊いろいろあるアニメ化の手法

アニメーションの基本は絵を描き，それを消したら別の位置へまた絵を描き，消す……ということのくりかえしになります．アニメーションに使えるものとしては

○キャラクタの組合せ（文字列操作）

○CIRCLEやLINEなどのグラフィック命令

○GET@，PUT@

○カラーパレット操作

などがあります．どのようなアニメにするかにより使う命令は変わってきますが，絵を描く時間が長いとアニメには見えないので，「描く・消す」が瞬時に行なわれなければいけません．

※XOR（エクスクルーシブ・オア）で重ねた絵は，カラーパレットが0になるので，見かけ上消えてしまう．もういちど重ねると，つぎは元の色にもどるので，アニメーションの強力な武器になる

F-BASICにはPUT@やLINEなどの命令にXORなどが使えるため，絵を消すのも合理的ですから，アニメの大半はGET@，PUT@で処理でき，時間的にも早く，有利です．

＊＊ロケットと4つの姿勢の作りかた

アニメの手始めとして，小さなロケットを作り，それをPUT@して，画面上で動かしましょう．このプログラムではロケットを4つの姿勢にして，4つのカーソル移動キーにより，上下左右に進ませています．まずロケットをひとつ，上向きにして作ります．座標は数学的なものにし，Y軸に対称なものとしました．平面図形の回転と同じ手法で，0°，90°，180°，270°とそれぞれの絵を描き，G1～G4の配列にGET@します．

ロケットはまずXS，YSの座標を基準にして，上向きのものを出力し（行番号210），カーソルキーによってそれぞれのルーチンへ飛ぶようにしています．なお1回のキー入力で，X方向には8ドット，Y方向へは4ドット分進むようにしていますが，スピード感を増したかったら，このドット数を変えてみましょう．

なお，このロケットは燃料噴射のパターンを赤でペイントしていますが，回転計算と共にペイント位置が変わるので注意深くプログラムを作る必要があります．ここではペイント位置指定用に独立した座標を用意しました．ロケットの形などに工夫をしたり，これをもとにしてゲームを作ったりすると，さらに楽しいものになります．

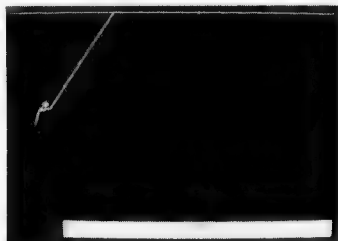
```
100 '============================ ﾛｹｯﾄ =======================================
110 '                   *** ｶｰｿﾙ KEY ﾃﾞ ﾛｹｯﾄｦ 4 ﾎｳｺｳﾆ ｺﾝﾄﾛｰﾙ***
120 DIM G1%(130),G2%(130),G3%(130),G4%(130),X(21),Y(21),XR(21),YR(21)
130 PAI=3.14159:SY=.43:X0=200:Y0=100:XS=300:YS=150:WIDTH80,25:CLS
140 FOR I=1 TO 21:READ X(I),Y(I):NEXT:'......................ﾛｹｯﾄ ｻﾞﾋｮｳﾉ ﾖﾐｺﾐ
150 '--------------------- 4ﾂﾉ ｶｲﾃﾝ ｽﾞｹｲｦ ﾂｸﾘ GET@ ｽﾙ --------------------------
160 RAD=0:GOSUB 450:GOSUB 460:GET@A(X0-12,Y0)-(X0+12,Y0+25),G1%,G:CLS:'UP
170 RAD=.5*PAI:GOSUB 450:GOSUB 460:GET@A(X0,Y0-5)-(X0+56,Y0+5),G2%,G:CLS:'L
180 RAD=PAI:GOSUB 450:GOSUB 460:GET@A(X0-12,Y0)-(X0+12,Y0-25),G3%,G:CLS:'DOWN
190 RAD=1.5*PAI:GOSUB 450:GOSUB 460:GET@A(X0-56,Y0-5)-(X0,Y0+5),G4%,G:CLS:'R
200 '-------------------------- ﾛｹｯﾄﾉ ｿｳｻ ------------------------------------
210 PUT@A(XS,YS)-(XS+24,YS+25),G1%,PSET
220 SEIGYO$=INKEY$:IF SEIGYO$="" THEN 220
230 IF SEIGYO$=CHR$(&H1E) THEN GOSUB 440:GOTO 280:'UP
240 IF SEIGYO$=CHR$(&H1F) THEN GOSUB 440:GOTO 320:'DOWN
250 IF SEIGYO$=CHR$(&H1C) THEN GOSUB 440:GOTO 360:'RIGHT
260 IF SEIGYO$=CHR$(&H1D) THEN GOSUB 440:GOTO 400:'LEFT
270 '**** UP
280 PUT@A(XS,YS)-(XS+24,YS+25),G1%,PSET:PUT@A(XS,YS)-(XS+24,YS+25),G1%,XOR
290 YS=YS-4:IF YS<30 THEN YS=30:'...........................ﾂｷﾞﾉ ｲﾁｦ ｹｲｻﾝ
300 IF INKEY$=CHR$(&H1E) THEN 280 ELSE  PUT@A(XS,YS)-(XS+24,YS+25),G1%,PSET:GOTO
 220
310 '**** DOWN
320 PUT@A(XS,YS-25)-(XS+24,YS),G3%,PSET:PUT@A(XS,YS-25)-(XS+24,YS),G3%,XOR
330 YS=YS+4:IF YS>160 THEN YS=160:'.........................ﾂｷﾞﾉ ｲﾁｦ ｲﾁｹｲｻﾝ
340 IF INKEY$=CHR$(&H1F) THEN 320 ELSE  PUT@A(XS,YS-25)-(XS+24,YS),G3%,PSET:GOTO
 220
350 '**** RIGHT
360 PUT@A(XS,YS)-(XS+56,YS+10),G4%,PSET:PUT@A(XS,YS)-(XS+56,YS+10),G4%,XOR
370 XS=XS+8:IF XS>580 THEN XS=580:'.........................ﾂｷﾞﾉ ｲﾁｦ ｹｲｻﾝ
380 IF INKEY$=CHR$(&H1C) THEN 360 ELSE  PUT@A(XS,YS)-(XS+56,YS+10),G4%,PSET:GOTO
 220
390 '**** LEFT
400 PUT@A(XS,YS)-(XS+56,YS+10),G2%,PSET:PUT@A(XS,YS)-(XS+56,YS+10),G2%,XOR
410 XS=XS-8:IF XS<60 THEN XS=60:'...........................ﾂｷﾞﾉ ｲﾁｦ ｹｲｻﾝ
420 IF INKEY$=CHR$(&H1D) THEN 400 ELSE  PUT@A(XS,YS)-(XS+56,YS+10),G2%,PSET:GOTO
 220
430 '--------------------------- SUBROUTINES ---------------------------------
440 LINE(XS-56,YS-25)-(XS+56,YS+25),PSET,0,BF:RETURN:'...............ｽﾞｹｲﾉ ｼｮｳｷｮ
450 FOR I=1 TO 21:XR(I)=X(I)*COS(RAD)-Y(I)*SIN(RAD):YR(I)=X(I)*SIN(RAD)+Y(I)*COS
(RAD):XR(I)=X0+XR(I):YR(I)=Y0-SY*YR(I):NEXT:RETURN:'.................ｶｲﾃﾝ ｹｲｻﾝ
460 FOR I=1 TO 12:CONNECT(XR(I),YR(I))-(XR(I+1),YR(I+1)),7:NEXT:'...ﾛｹｯﾄ
470 FOR I=14 TO 19:CONNECT(XR(I),YR(I))-(XR(I+1),YR(I+1)),2:NEXT:PAINT(XR(21),YR
(21)),2:RETURN:'.....................................................ﾌﾝｼｬﾄ PAINT
480 DATA  0,0,    6,-12,      6,-32,  12,-38,  12,-48,  5,-42, 0,-40:'ﾛｹｯﾄ
490 DATA  -5,-42, -12,-48,  -12,-38,  -6,-32,  -6,-12,  0,0:'          ﾛｹｯﾄ
500 DATA  0,-40,  5,-45,    5,-50,    0,-55,   -5,-50, -5,-45, 0,-40:'ﾌﾝｼｬ
510 DATA  0,-43                                                     :'PAINT ｲﾁ
```

空中ブランコで遊ぶ

＊5つの絵によるアニメでも，かなりのものに

絵の数は少なくても，ディスプレイ方法によってはけっこうそれらしいアニメに見える例を紹介しましょう．空中ブランコをする人が，キー入力と共に空中に飛出し，目的の位置にうまく着地できるかどうかを楽しむものです．

例によってブランコにつかまっている人間のパターンを描き，それをGET@します．大きなパターンをGET@，PUT@すると不自然に見えることがありますが，このプログラムのように55×25ドット程度のものでは瞬時に描き，瞬時に消えますのでアニメとして十分に通用します．5つのパターンをGET@したのちに，天井と床，そして着地の目標（乱数を使って，出力位置を変えています）などを描きます．なお，PFキーの5を押すと，人間がブランコから飛出すようになっているため，割込みの文も書きます（行番号240）．

ブランコはさいしょ，ゆっくりと往復し，時間と共にだんだん速くなるようにするために，ここでは変数（SPEED）にまず600を入れています．5つのパターンはそれぞれ5つのサブルーチンによって呼出し，行きと帰りで，出力順序を変えています（行番号260，270）．1往復したら，速度を変えますが，ある速さになったらその変化率を変えるようにしました（行番号280）．なお，ある速さになったらまたゆっくりと動くようにして，ただ見ているだけでも飽きないようにしてあります（行番号290）．

PFキーが押され，それが前へ飛出す時（変数Fで判断）には，ブランコの速さによって着地点を変える計算をし（行番号470，480），設定範囲内で空中を進みます（行番号500〜560）．ここでは着地の成功・失敗の判断はしていませんが，ゲームを作ったり，空中の姿勢に工夫をして，ウルトラCの演技をさせるのもよいでしょう．

```
100 '========================== ｸｳﾁｭｳ ﾌﾞﾗﾝｺ ====================================
110 DEFINT A-Z:GX=55:GY=25:H=INT(GX*GY/16)+8:DIM G1(H),G2(H),G3(H),G4(H),G5(H)
130 SCREEN 7,0:CLS:'·································SCREEN 7,0 ﾃﾞ ｶﾞﾒﾝｦ ｹｼﾃｵｸ
140 CIRCLE(20,5),5,7,,,,F:CONNECT(20,5)-(7,11)-(0,23):CONNECT(10,10)-(27,9):GET@
(0,0)-(GX,GY),G1,G,7:CLS:'···································MODE1
150 CIRCLE(20,5),5,7,,,,F:CONNECT(20,5)-(13,14)-(10,25):CONNECT(15,11)-(32,9):GE
T@(0,0)-(GX,GY),G2,G,7:CLS:'·································MODE2
```

```
160 CIRCLE(20,5),5,7,,,,F:CONNECT(20,5)-(20,26):CONNECT(20,12)-(35,8):GET@(0,0)-
(GX,GY),G3,G,7:CLS:'･････････････････････････････････････････MODE3
170 CIRCLE(20,5),5,7,,,,F:CONNECT(20,5)-(29,14)-(50,20):CONNECT(26,11)-(42,8):GE
T@(0,0)-(GX,GY),G4,G,7:CLS:'･････････････････････････････････MODE4
180 CIRCLE(20,5),5,7,,,,F:CONNECT(20,5)-(34,13)-(54,17):CONNECT(29,10)-(43,6):GE
T@(0,0)-(GX,GY),G5,G,7:CLS:'･････････････････････････････････MODE5
190 '========= ﾃﾞｨｽﾌﾟﾚｲ
200 IF TIME>30000 THEN TIME$="00:00:00"
210 RANDOMIZE TIME:SCREEN 7,7:WIDTH 40,25:CLS
220 LINE(0,0)-(639,0),PSET,4:LINE(100,185)-(639,199),PSET,4,BF
230 XRND=INT(RND*220)+260:LINE(XRND,185)-(XRND+30,193),PSET,2,BF:'･･････ﾁｬｸﾁﾀﾞｲ
240 ONKEY(5) GOSUB 440:KEY(5) ON:'･････････････････････････････PF KEYﾉ ﾜﾘｺﾐ
250 SPEED=600
260  F=1: GOSUB 310:GOSUB 330:GOSUB 350:GOSUB 370:GOSUB 390:'･････ﾕｷ
270  F=-1: GOSUB 390:GOSUB 370:GOSUB 350:GOSUB 330:GOSUB 310:'････ｶｴﾘ
280 IF SPEED<=200 THEN SPEED=SPEED-5 ELSE SPEED=SPEED-50
290 IF SPEED<=50 THEN  250 ELSE 260
300 '========= PUT@ SUB
310 M=1:X=50:Y=75:LINE(200,0)-(79,84),PSET,6:PUT@(X,Y)-(X+GX,Y+GY),G1,PSET:GOSUB
 420
320 LINE(200,0)-(79,84),PRESET:PUT@(X,Y)-(X+GX,Y+GY),G1,PRESET:RETURN
330 M=2:X=97:Y=94:LINE(200,0)-(130,102),PSET,6:PUT@(X,Y)-(X+GX,Y+GY),G2,PSET:GOS
UB 420
340 LINE(200,0)-(130,102),PRESET:PUT@(X,Y)-(X+GX,Y+GY),G2,PRESET:RETURN
350 M=3:X=163:Y=103:LINE(200,0)-(200,110),PSET,6:PUT@(X,Y)-(X+GX,Y+GY),G3,PSET:G
OSUB 420
360 LINE(200,0)-(200,110),PRESET:PUT@(X,Y)-(X+GX,Y+GY),G3,PRESET:RETURN
370 M=4:X=225:Y=94:LINE(200,0)-(270,102),PSET,6:PUT@(X,Y)-(X+GX,Y+GY),G4,PSET:GO
SUB 420
380 LINE(200,0)-(270,102),PRESET:PUT@(X,Y)-(X+GX,Y+GY),G4,PRESET:RETURN
390 M=5:X=280:Y=79:LINE(200,0)-(321,84),PSET,6:PUT@(X,Y)-(X+GX,Y+GY),G5,PSET:GOS
UB 420
400 LINE(200,0)-(321,84),PRESET:PUT@(X,Y)-(X+GX,Y+GY),G5,PRESET:RETURN
420 FOR T=0 TO SPEED:NEXT:RETURN:'･･････････････････････････････ｼﾞｶﾝﾁｮｳｾｲ
440 ON M GOSUB 320,340,360,380,400:'･･････････････････････PF KEY ｶﾞ ｵｻﾚﾀﾄｷﾉ ｼｮﾘ
450 IF F>=0 THEN 470 ELSE 590:'･･･････････････････････････････ﾎｳｺｳﾉ ﾊﾝﾀﾞﾝ
460 '
470 YM=INT(6000/SPEED):IF YM>50 THEN YM=50:'･･･････････ﾌﾞﾗﾝｺﾉ ﾊﾔｻﾄ ﾄﾌﾞｷｮﾘﾉ ｹｲｻﾝ
480 XM=INT(12000/SPEED):IF XM>80 THEN XM=80:'･･････････ﾌﾞﾗﾝｺﾉ ﾊﾔｻﾄ ﾄﾌﾞｷｮﾘﾉ ｹｲｻﾝ
490 X=X+XM:Y=Y-YM
500 WHILE X<540 AND Y<160
510    PUT@(X,Y)-(X+GX,Y+GY),G5,PSET:FOR T=0 TO INT(5000/(Y*50)):NEXT
520    LINE(200,0)-(200,110),PSET,6:'････････････････････････ﾂﾅｦ ﾏｼﾀﾆ ﾓﾄﾞｽ
530    PUT@(X,Y)-(X+GX,Y+GY),G5,PRESET
540    X=X+10:Y=Y+5
550 WEND
560 PUT@(X+20,160)-(X+20+GX,160+GY),G3,PSET:'･････････････････ﾁｬｸﾁ
580 FOR T=0 TO 9000:NEXT:KEY(5) OFF:GOTO 200:'･････････････････ﾓｳｲﾁﾄﾞ PLAY
590 BEEP:LOCATE 16,20:PRINT"NG!!!!!!!":GOTO 580
```

24枚の組合せゲーム

＊15ゲームを基本にして

オモチャ屋，文房具店などで見かける15ゲームは，４×４枚のチップに１～15までの数字が書かれ，16枚のチップのうち１枚を抜いてから自由に動かし，また元にもどして数字をそろえたり，異なる順序に並べかえたりするものです．25枚のチップを作り，そこに絵を描きGET@，PUT@を使って元どおりにするものを考えましょう．

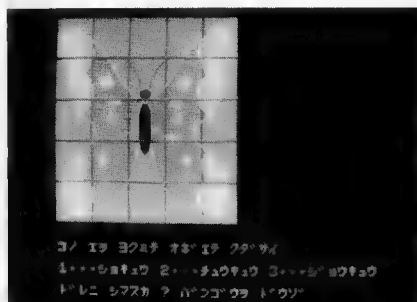
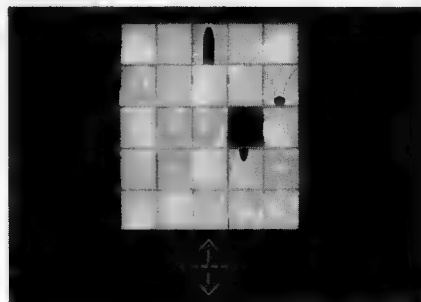
数字とは違って絵の場合はより難かしくなりますし，似たようなパターンがあるとジグソーパズルと同じで，どの位置に動かせばよいのかがすぐには分らないという面白さも出てきます．15ゲームよりも枚数が多いだけに，真剣に取組んでもなかなか完成しません．

＊＊25個分をGET@，PUT@する

ここで取上げた絵の素材は蝶々です．Ｙ軸を中心にして左右対称の図形とし，作りやすくしたこと，それぞれのチップによく似た模様があるので，まごつきやすいことなどから採用しています．幾何学的な模様にすればもっと難しくなります．

なお形式４の，フルカラーでGET@する時，画面いっぱいの絵ではメモリが不足します．今回取上げた大きさがギリギリといえるでしょう．なお，蝶々の原図のデータに対し，画面に描くものは縮率をかけて少し小さくしています．変数（SX，SY）の値を変えたり，ひとつのチップの寸法（XW，YW）を変えれば自由なサイズのものを作ることができます．このプログラムでは赤色の枠を作り，その中に絵を描いています．

＊＊＊ゲームのアルゴリズムについて

15ゲームそのものの作りかたもいろいろなアルゴリズムが考えられます．ここでは鮫島正裕氏によるもの（ラジオ技術社刊，パソコン実用ガイド'83所載）を参考にし，チップ数を増やしたものにしています．５×５のそれぞれのマス目に１～25までの数値を与え，カーソルキーを使って動かすときの位置によって数値を交換します．空いている場所は26を与えて，サブルーチンでその場所を消すようにしています．この手法のほかに，上下に動いたときは５ずつ変化し，左右に動いたときは１ずつの変化であることを利用したチップの移動方法もあります．長いリストですが，基本はやさしいのでじっくり見て下さい．

◀GET@する画面

ゲーム中の画面▶

```
10 '*************************** GET@/PUT@ ｴｱﾜｾ ﾊﾟｽﾞﾙ ***************************
20 '
30     DEFINT A-Z:WIDTH 40,25:CLS
40 SX!=.78:SY!=.8:DIM X(35),Y(35):X0=180*SX!:XW=56:YW=28:GOSUB 950:ERASE X,Y
50     REM SX/SY:ｼｭｸﾘﾂ  XW/YW:ﾁｯﾌﾟｻｲｽﾞ  X(I)/Y(I):ﾁｮｳﾁｮﾉ ｻﾞﾋｮｳ   X0:ﾀｲｼｮｳｼﾞｸ
60 GOSUB 1510:'..............................................................GET@ SUB
70   LOCATE 0,20:PRINT"ｺﾉ ｴｦ ﾖｸﾐﾃ ｵﾎﾞｴﾃ ｸﾀﾞｻｲ";
80   LOCATE 0,22:PRINT"1･･･ｼｮｷｭｳ 2･･･ﾁｭｳｷｭｳ 3･･･ｼﾞｮｳｷｭｳ"
90   LOCATE 0,24:PRINT"ﾄﾞﾚﾆ ｼﾏｽｶ ? ﾊﾞﾝｺﾞｳｦ ﾄﾞｳｿﾞ";
100 G$=INPUT$(1):G=VAL(G$):IF G<1 OR G>3 THEN BEEP:GOTO 100
110 GRADE=G*G*40:'.............................................ﾅﾗﾍﾞｶｴﾉ ｶｲｽｳ
120 '
130 DIM A(4,4):CLS
140 'ﾊﾞﾝｺﾞｳﾉ ﾜﾘｱﾃ
150 FOR GYO=0 TO 4:FOR RETSU=0 TO 4:PICT(GYO,RETSU)=GYO*5+(RETSU+1):NEXT:NEXT
160 PICT(4,2)=26:'.......................................ﾇｷﾄｯﾀ 1ﾏｲﾉ ﾁｯﾌﾟﾉ ｼﾃｲ
170 FOR GY=0 TO 4:FOR RE=0 TO 4:GOSUB 600:NEXT RE:NEXT GY:'.........ﾓﾄﾉ ｴｦ ﾀﾞｽ
180 GY=4:RE=2
190 'ﾃﾞﾀﾗﾒﾆ ﾅﾗﾍﾞﾙ
200     IF TIME>30000 THEN TIME$="00:00:00"
210     RANDOMIZE TIME
220 FOR KAISU=1 TO GRADE:RAND=INT(RND*4)+1:ON RAND GOSUB 420,460,500,540:NEXT
230 'ｱｲﾃｲﾙ ﾊﾞｼｮｦ ｻｶﾞｽ
240 FOR GYO=0 TO 4:FOR RETSU=0 TO 4
250  IF PICT(GYO,RETSU)=26 THEN GY=GYO:RE=RETSU:GOTO 280'......ｱｲﾃｲﾙ ﾊﾞｼｮｦ ﾊｯｹﾝ
260 NEXT RETSU,GYO
270 '
280 LOCATE 17,19:PRINT.      "  /\  ";
290 LOCATE 17,20:PRINT       "   |   ";
300 LOCATE 17,21:PRINT       "<---->";
310 LOCATE 17,22:PRINT       "   |   ";
320 LOCATE 17,23:PRINT       "  \/  ";
330 '
340 'ﾆｭｳﾘｮｸ
350 A$=INKEY$:IF A$="" THEN 350
360   MOVE=32-ASC(A$)
370 IF MOVE<1 OR MOVE>4 THEN 350
```

```
380   ON MOVE GOSUB 420,460,500,540
390 GOTO 350
400 '
410 'UP
420 IF GY=0 THEN BEEP:RETURN
430    GOSUB 580:SWAP PICT(GY,RE),PICT(GY-1,RE):GOSUB 600:GY=GY-1:GOSUB 600
440 RETURN
450 'DOWN
460 IF GY=4 THEN BEEP:RETURN
470    GOSUB 580:SWAP PICT(GY,RE),PICT(GY+1,RE):GOSUB 600:GY=GY+1:GOSUB 600
480 RETURN
490 'RIGHT
500 IF RE=4 THEN BEEP:RETURN
510    GOSUB 580:SWAP PICT(GY,RE),PICT(GY,RE+1):GOSUB 600:RE=RE+1:GOSUB 600
520 RETURN
530 'LEFT
540 IF RE=0 THEN BEEP:RETURN
550    GOSUB 580:SWAP PICT(GY,RE),PICT(GY,RE-1):GOSUB 600:RE=RE-1:GOSUB 600
560 RETURN
570 '
580 PLAY"V10T250O5L16GL4CDE":RETURN
590 '
600 NUM=PICT(GY,RE):XS=180+RE*(XW+1):YS=GY*(YW+1):'.............ﾁｯﾌﾟﾉ ｵｸｲﾁﾉ ｹｲｻﾝ
610 ON NUM GOSUB 640,650,660,670,680,690,700,710,720,730,740,750,760,770,780,790
,800,810,820,830,840,850,860,870,880,900
620 RETURN
630 '
640 PUT@A(XS,YS)-(XS+XW,YS+YW),G1,PSET:RETURN
650 PUT@A(XS,YS)-(XS+XW,YS+YW),G2,PSET:RETURN
660 PUT@A(XS,YS)-(XS+XW,YS+YW),G3,PSET:RETURN
670 PUT@A(XS,YS)-(XS+XW,YS+YW),G4,PSET:RETURN
680 PUT@A(XS,YS)-(XS+XW,YS+YW),G5,PSET:RETURN
690 PUT@A(XS,YS)-(XS+XW,YS+YW),G6,PSET:RETURN
700 PUT@A(XS,YS)-(XS+XW,YS+YW),G7,PSET:RETURN
710 PUT@A(XS,YS)-(XS+XW,YS+YW),G8,PSET:RETURN
720 PUT@A(XS,YS)-(XS+XW,YS+YW),G9,PSET:RETURN
730 PUT@A(XS,YS)-(XS+XW,YS+YW),G10,PSET:RETURN
740 PUT@A(XS,YS)-(XS+XW,YS+YW),G11,PSET:RETURN
750 PUT@A(XS,YS)-(XS+XW,YS+YW),G12,PSET:RETURN
760 PUT@A(XS,YS)-(XS+XW,YS+YW),G13,PSET:RETURN
770 PUT@A(XS,YS)-(XS+XW,YS+YW),G14,PSET:RETURN
780 PUT@A(XS,YS)-(XS+XW,YS+YW),G15,PSET:RETURN
790 PUT@A(XS,YS)-(XS+XW,YS+YW),G16,PSET:RETURN
800 PUT@A(XS,YS)-(XS+XW,YS+YW),G17,PSET:RETURN
810 PUT@A(XS,YS)-(XS+XW,YS+YW),G18,PSET:RETURN
820 PUT@A(XS,YS)-(XS+XW,YS+YW),G19,PSET:RETURN
830 PUT@A(XS,YS)-(XS+XW,YS+YW),G20,PSET:RETURN
840 PUT@A(XS,YS)-(XS+XW,YS+YW),G21,PSET:RETURN
850 PUT@A(XS,YS)-(XS+XW,YS+YW),G22,PSET:RETURN
```

```
860 PUT@A(XS,YS)-(XS+XW,YS+YW),G23,PSET:RETURN
870 PUT@A(XS,YS)-(XS+XW,YS+YW),G24,PSET:RETURN
880 PUT@A(XS,YS)-(XS+XW,YS+YW),G25,PSET:RETURN
890 '
900 LINE(XS,YS)-(XS+XW,YS+YW),PSET,2,B:PAINT(XS+2,YS+2),0,2:RETURN
910 '
920 '=========================== DRAW & GET@ SUB ============================
930 '**************************** ﾁｮｳﾁｮ ｦ ｴｶﾞｸ ********************************
940 'ｱﾀﾏ
950 RESTORE 960:C=7:GOSUB 1440
960 DATA 6, 180,62, 168,64,167,68, 169,71, 172,73, 180,75
970 'ｼｮｯｶｸ
980 RESTORE 990:C=1:GOSUB 1440
990 DATA 6, 132,36,148,41,157,46,165,52,169,56,171,63
1000 'ｳｴﾉ ﾊﾈ
1010 RESTORE 1020:C=7:GOSUB 1440
1020 DATA 35, 172,73, 150,65,130,55,110,40,90,28,70,20,50,15,40,10,30,9,20,10,13
,14,8,19,7,26,7,30,9,35,17,39,25,42,24,50,26,61,30,70,37,78,48,83,56,88,68,91,78
,93,80,97,83,104,92,107,101,108,117,106,131,105,143,102,154,100,162,98,166,96
1030 'ｼﾀﾉ ﾊﾈ
1040 RESTORE 1050:C=7:GOSUB 1440
1050 DATA 30,  90,106,70,111,52,118,37,125,23,132,15,138,13,143,15,148,19,155,21
,161,21,167,23,170,27,173,35,174,46,175,56,176,66,175,76,172,84,169,87,168,107,1
65,119,160,131,155,139,150,147,144,154,137,160,125,164,118,166,114,167,107
1060 'ﾄﾞｳﾀｲ
1070 RESTORE 1080:C=7:GOSUB 1440
1080 DATA 10,  175,74,169,80,167,86,167,105,168,111,171,116,173,119,175,120,178,
121,180,121
1090 'ｳｴﾉ ﾊﾈﾉ ﾓﾖｳ1
1100 RESTORE 1110:C=7:GOSUB 1440
1110 DATA 14,  72,21,73,28,75,33,72,38,67,43,62,47,54,51,70,55,75,60,77,64,76,69
,72,75,64,80,53,85
1120 'ｳｴﾉ ﾊﾈﾉ ﾓﾖｳ2
1130 RESTORE 1140:C=7:GOSUB 1440
1140 DATA 11,  100,53,111,55,117,58,115,60,111,61,103,62,95,60,91,58,89,55,93,53
,100,53
1150 'ｳｴﾉ ﾊﾈﾉ ﾓﾖｳ3
1160 RESTORE 1170:C=7:GOSUB 1440
1170 DATA 11,  106,72,111,70,119,70,126,72,130,74,127,76,119,77,112,76,108,75,10
6,74,106,72
1180 'ｳｴﾉ ﾊﾈﾉ ﾓﾖｳ4
1190 RESTORE 1200:C=7:GOSUB 1440
1200 DATA 10,  105,89,107,87,114,85,121,86,125,88,122,91,116,92,111,91,107,90,10
5,89
1210 'ｼﾀﾉ ﾊﾈﾉ ﾓﾖｳ1
1220 RESTORE 1230:C=7:GOSUB 1440
1230 DATA 17,  50,119,58,121,64,124,67,129,63,135,58,139,52,143,62,145,72,149,77
,152,80,155,77,159,73,164,67,168,64,169,59,173,55,176
1240 'ｼﾀﾉ ﾊﾈﾉ ﾓﾖｳ2
```

```
1250 RESTORE 1260:C=7:GOSUB 1440
1260 DATA 9,  92,123,99,122,104,124,105,126,102,128,97,129,90,127,89,124,92,123
1270 'ｼﾀﾉ ﾊﾈﾉ ﾓﾖｳ3
1280 RESTORE 1290:C=7:GOSUB 1440
1290 DATA 8,  95,143,100,142,107,143,110,145,106,148,99,148,94,145,95,143
1300 'PAINT
1310 PAINT(27,43),6,7:PAINT(2*X0-27,43),6,7
1320 PAINT(78,45),3,7:PAINT(2*X0-78,45),3,7
1330 PAINT(92,59),3,7:PAINT(2*X0-92,59),3,7
1340 PAINT(92,71),3,7:PAINT(2*X0-92,71),3,7
1350 PAINT(76,100),3,7:PAINT(2*X0-76,100),3,7
1360 PAINT(79,116),3,7:PAINT(2*X0-79,116),3,7
1370 PAINT(24,120),6,7:PAINT(2*X0-24,120),6,7
1380 PAINT(59,68),1,7:PAINT(2*X0-59,68),1,7
1390 PAINT(59,95),1,7:PAINT(2*X0-59,95),1,7
1400 LINE(0,0)-(5*(XW+1),5*(YW+1)),PSET,2,B:PAINT(1,1),4,7,1,2
1410 'ﾜｸ
1420 FOR X=0 TO 4*(XW+1) STEP XW+1:FOR Y=0 TO 4*(YW+1) STEP YW+1:LINE(X,Y)-(X+XW
,Y+YW),PSET,2,B:NEXT:NEXT:RETURN
1430 'ﾁｮｳﾁｮ ﾉ ｻﾞﾋｮｳ ｹｲｻﾝ
1440 READ N:FOR P=1 TO N:READ X(P),Y(P):X(P)=X(P)*SX!:Y(P)=Y(P)*SY!:NEXT:FOR P=1
 TO N-1:GOSUB 1480:NEXT
1450 'ﾐｷﾞﾊﾝﾌﾞﾝﾉ ﾀｲｼｮｳ ｽﾞｹｲﾉ ｻﾞﾋｮｳｦ ｹｲｻﾝ
1460 FOR P=1 TO N:X(P)=2*X0-X(P):NEXT:FOR P=1 TO N-1:GOSUB 1480:NEXT:RETURN
1470 'ﾁｮｳ ｦ ｴｶﾞｸ
1480 CONNECT (X(P),Y(P))-(X(P+1),Y(P+1)),C:RETURN
1490 '
1500 '************************** 25 ﾊﾟﾀｰﾝﾉ GET@**********************************
1510 DIM G1(310),G2(310),G3(310),G4(310),G5(310),G6(310),G7(310),G8(310),G9(310)
,G10(310),G11(310),G12(310),G13(310),G14(310),G15(310),G16(310),G17(310),G18(310
),G19(310),G20(310),G21(310),G22(310),G23(310),G24(310),G25(310)
1520 XS=0:YS=0:GET@A(XS,YS)-(XS+XW,YS+YW),G1,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS
+YW),G2,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G3,G:GOSUB 1570:GET@A(XS,YS)-(XS
+XW,YS+YW),G4,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G5,G
1530 XS=0:YS=YS+YW+1:GET@A(XS,YS)-(XS+XW,YS+YW),G6,G:GOSUB 1570:GET@A(XS,YS)-(XS
+XW,YS+YW),G7,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G8,G:GOSUB 1570:GET@A(XS,Y
S)-(XS+XW,YS+YW),G9,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G10,G
1540 XS=0:YS=YS+YW+1:GET@A(XS,YS)-(XS+XW,YS+YW),G11,G:GOSUB 1570:GET@A(XS,YS)-(X
S+XW,YS+YW),G12,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G13,G:GOSUB 1570:GET@A(X
S,YS)-(XS+XW,YS+YW),G14,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G15,G
1550 XS=0:YS=YS+YW+1:GET@A(XS,YS)-(XS+XW,YS+YW),G16,G:GOSUB 1570:GET@A(XS,YS)-(X
S+XW,YS+YW),G17,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G18,G:GOSUB 1570:GET@A(X
S,YS)-(XS+XW,YS+YW),G19,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G20,G
1560 XS=0:YS=YS+YW+1:GET@A(XS,YS)-(XS+XW,YS+YW),G21,G:GOSUB 1570:GET@A(XS,YS)-(X
S+XW,YS+YW),G22,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G23,G:GOSUB 1570:GET@A(X
S,YS)-(XS+XW,YS+YW),G24,G:GOSUB 1570:GET@A(XS,YS)-(XS+XW,YS+YW),G25,G
1570 XS=XS+XW+1:RETURN:'..............................Xｻﾞﾋｮｳｦ ｼﾞｭﾝﾊﾞﾝﾆ ﾌﾔｼﾃｲｸ
```

6
漢字の取扱いと
その応用

漢字を画面に出してみる

＊一覧表をまず見ましょう

FM-77にはJISに定められた，第1水準の漢字をROMの形で内蔵しています．漢字のそれぞれにはコードが割付けられていますが，呼出しは4ケタの16進数を使うことが基本になっており，マニュアルの巻末などにコードと共に一覧表の形で掲載されています．10進数を使っても呼出せることはいうまでもありません．

一覧表を見れば分ることですが，漢字は音読みを基本にしていること，またコードは&H2120～&H4F53（10進数で8480～20307）あたりまで付けられていますが，その間がビッシリと埋められてないことが分ります．漢字の出力は，もっとも簡単な例では

PRINT@，コード

という文になります．漢字出力はグラフィック画面を使うこと，またプリンタへの出力にも制限があることなどをまず理解しましょう．

※JISでは区，点とも1～94に設定されている．FM-77では，点については&H21～&H7E，区については&H21～&H4Fとしている．たとえば「澄」はJISでは32区1点であるが，FM-77のコード体系では第1バイト（区）を&H40，第2バイト（点）を&H21としており，&H4021で「澄」を指定することになる

＊＊いろいろと分る，一覧表の内容

なにはともあれ，まず漢字を画面に出してみます．サンプルプログラムを見てください．ここでは画面に出力する漢字のコードを入力すると，1行あたり16文字，そして8行にわたって漢字を出力します．コード体系の基本は本来なら下2ケタは0，1，2，…，E，Fとな

```
100 '====================== ｶﾝｼﾞ ｲﾁﾗﾝ ｼｭﾂﾘｮｸ ===========================
110 DEFINT A-Z:X0=16:Y0=8:WIDTH80,25:CLS
120 LINE(0,0)-(2*X0+16*24,20*8+10),PSET,6,B
130 LOCATE 0,22:INPUT"START/ ｶﾝｼﾞ CODE  ";ST$:ST=VAL("&H"+ST$)
140 '
150 FOR GYO=0 TO 7
160     FOR MOJI=0 TO 15
170         KCODE=ST+(16*GYO+MOJI)
180            PRINT@(X0+MOJI*24,Y0+GYO*20),KCODE
190     NEXT MOJI
200 NEXT  GYO
205 '
210 LOCATE 0,23:PRINT"ﾂｷﾞｦ ﾐﾙﾅﾗ PUSH ANY KEY";
220 WHILE INKEY$="":WEND
230 ST=KCODE+1:LOCATE 0,22:PRINT USING"ｽﾀｰﾄ CODE=&   &           ";HEX$(ST)
240 LINE(1,1)-(2*X0+16*24-1,20*8+10-1),PSET,0,BF
250 GOTO 150
```

```
　押旺横欧殴王翁襖鴬鴎黄岡沖荻億
屋憶臆桶牡乙俺卸恩温穏音下化仮何
伽価佳加可嘉夏嫁家寡科暇果架歌河
火珂禍禾稼箇花苛茄荷華菓蝦課嘩貨
迦過霞蚊俄峨我牙画臥芽蛾賀雅餓駕
介会解回塊壊廻快怪悔恢懐戒拐改

START/ ｶﾝｼﾞ CODE  ? 3220
ﾂｷﾞｦ ﾐﾙﾅﾗ PUSH ANY KEY
```

HARDC2によってコピーした一覧表▶

るはずですが，一覧表では2，4，5，6，7しか使っていません．したがってサンプルプログラムで漢字を出力すると，空白が出てきますが，とりあえずこれを動かしてみて漢字出力に慣れましょう．

まず気がつくことは，違うコードを入力しても，出てくる漢字は同じになることです．たとえば「亜」は&H3021，&H30A1の両方が使えます．また，ひらがなとカタカナは，&HFF（256）分離れたコードになっていますが，カタカナには「ヴ，ヵ，ヶ」の3文字が余計にあることです．ひらがなをカタカナに，カタカナをひらがなに直すとき，ひらがな→カタカナ変換は可能ですが，カタカナからひらがなへの変換は，もし「ヴ」などが含まれると不可能になります．また，キャラクタコード表でのカタカナの出現順と，JISの漢字（正しくは非漢字）の出現順も異なっています．したがってキーボードから入力するキャラクタに，ある数を加えて4ケタの16進数にすればたちまち画面に漢字として扱うひらがなやカタカナを出力する，というわけにはゆきません．

漢字の一覧表をながめていると，このほかにいろいろと楽しいことが分るのですが，それはさておき，漢字処理をいろいろと考えてみましょう．その基本となるのは漢字のコード体系と，FM-77の内部メモリと外部記憶装置……フロッピーを使うことです．

読みから漢字を探す——1

＊読みの先頭を配列に入れる方法がある

まず漢字の呼出しを考えましょう．いちばんやさしいのは漢字一覧表にしたがって，ア，イ，……ワまでの漢字の先頭のコードを使う方法です．たとえばY＄(I)という配列を用意し

Y＄(1)＝"ア　3021"：Y＄(2)＝"イ　304A"……

という形の文字列データにしておきます．探したい漢字の読みを入力したら，その読みと配列とを比較します．この比較は

①IF　読み＝LEFT$(Y$(I)，1)　THEN……

Y＄(I)の左から1番目（つまりア，イ，……です）と，入力した読みが同じなら，という書き方

②IF　INSTR(Y$(I)，読み)　THEN……

Y＄(I)に，入力した読みが含まれていれば，という書き方

といった方法が使えます．

こうして目的のコードを含むY＄(I)という配列が見つかったら，

CODE$＝RIGHT$(Y$(I)，4)

つまり，配列の右から4つの文字のつづりを取出して，コードとします．入力した読みが“イ”としたら，これでコードは“304A”になります．

＊＊先頭のコードが分れば，それに続く漢字が求められる

コードが304Aである，ということが分ったので，これを漢字のデータとして使えばよいので，たとえば

PRINT@(X, Y)，VAL("&H"＋CODE$)

とすれば，画面に“イ”の先頭にある漢字が出力されます．なお，VALは文字列を数値化する関数ですが，10進数にはそのまま対応しても，16進数の文字列表記である今のCODE＄には，そのままでは対応しません．CODE＄＝“304A”として

VAL(CODE$)……数値は304になってしまう

VAL("&H"＋CODE$)……数値は12362で，16進数の“304A”と正しく対応する

になります．

こうして，たとえば“イ”の先頭のコードが見つかれば，ここから始まる漢字の一覧を画面に出すのはすぐに分るでしょう．17個分なら

```
FOR I=0 TO 16:ST=コードの先頭+I
  PRINT @(I*20, 0), ST
NEXT
```

といった形にすればよいわけです(138ページのプログラム参照).

＊＊＊読みを細分化すればよいが……

この方法でも，ある程度実用にはなります．それぞれの読みの先頭のコードはア，イ，ウ，……ワまで44ありますから，配列もそれだけの数で足ります．この方法をよく見ると，たとえば“ネ”の読みのものなどは少ないので，漢字はすぐに見つかりますが，“シ”や“カ”などは漢字が多く，目的の漢字を探す作業がたいへんになります．

便宜的には先頭のコード，という方法ではなく，たとえば“ア”なら，3035の“圧”あたりを仮の先頭コードにしておき，画面上では“圧”から始まる漢字を出力し，プログラムの方で，この“圧”を中心にして，前にも(梓……亜)出力できるようにすればよいでしょう．また，読みを増やして

ア　3021　　アツ　3035　　イ　304A　　イツ　306F
ウ　3126　　ウツ　3135　　……

といったようにする方法もあります．細分化すれば目的の漢字は探しやすくなりますが，プログラムはそれだけ複雑になります．もうひとつの問題としては，JISの漢字コードは音読みですから，たとえば“アツ”で探すと「圧　斡　扱」はありますが,「暑　熱　厚」などはまったく別のコード体系になっていることです．細分化しても，探す手間は必らずしも減りません．

JISの漢字一覧表を基準にして漢字を探す方法は，けっきょくのところ，どの方法をとるにしても限度があります．使い手が一覧表をよく見て暗記をし，たとえば“嘘”は“キョ”にはなく，“ウソ”の一覧中にあるといったように対応しなければいけないのです．

一覧表に頼らない漢字の探しかた，ということになれば，自分で読みの辞書を作るしか方法はありません．言葉の上ではこのひとことですみますが，いざ実行する，ということになれば，かなり大変です．まず時間がかかること，経験をしていないために使用頻度が多い漢字とそうでないものの扱いが分らないこと（このために，使用頻度の少ない漢字が探しやすく，多いものをデータの最後に置いて探しにくくなったりする）などをあらかじめ考えないといけません．

読みから漢字を探す——2

＊もっともやさしい辞書を作る

漢字一覧表に頼らないとすれば，読みかたとそれに対応する漢字をデータの形で整理しなければいけません．さきほどの例とおなじ手法をとったプログラムで考えましょう．サンプルはごく簡単なもので，読みと共にそれに対応するコードを入力し，やはり配列の形でいったんFM-77のメモリに入れておきます．ここではとりあえず2,000個分の読みを扱います．

入力が終了したら，いま作成したデータをシーケンシャルファイルに書きこんでおきます．これでとりあえず使い手が考える読みと，それに対応する漢字コードが外部記憶されたので，漢字一覧表に頼らなくても目的の漢字が探せることになります．

ただしこの方法を実用化しよう，ということになると，もちろんこれだけではすみません．第一，1回の作業ですべての読みの入力はできないので，追加のプログラムや訂正，削除のプログラムがなければいけないでしょう．またデータの中から目的のものを探す方法をじょうずに考えないと時間がかかってしまいます．

```
100 '============== ﾖﾐﾉ ｼﾞｼｮｦ ﾂｸﾙ SUB PRO. ==============
110 '    *ﾌﾟﾛｸﾞﾗﾑ ﾌｧｲﾙﾒｲ=ｼﾞｼｮｻｸｾｲ  ﾄﾞﾗｲﾌﾞ=0
120 DEFINT A-Z
130 N=1:CLEAR 3000:DIM YOMI$(2000)
140 CLS
150 LOCATE 0,5:INPUT"ﾖﾐ  ";YOMI$
160   INPUT"ｺｰﾄﾞ  ";KCODE$:PRINT@(50,10),VAL("&H"+KCODE$)
170   YOMI$(N)=YOMI$+KCODE$
180   PRINT"ﾃｲｾｲ ･･･>T  ｵﾜﾘ････>E  ﾂｷﾞﾉﾆｭｳﾘｮｸ  ANY KEY"
190    YN$=INKEY$:IF YN$="" THEN 190
200 IF YN$="T" OR YN$="t" THEN 140
210 IF YN$="E" OR YN$="e" THEN 240
220 N=N+1:IF N>2000 THEN 240 ELSE GOTO 140
230 'ﾌｧｲﾙﾆ ｷﾛｸ
240 OPEN"O",#1,"ｼﾞｼｮ"
250 PRINT #1,N
260 FOR I=1 TO N
270  PRINT #1,YOMI$(I)
280 NEXT
290 CLOSE
```

読みを入力するプログラム▶

データの中から目的のものを▶
探すプログラム

```
100 '========== ｼﾞｼｮｦﾂｶｳ SUB PRO. ==============
110 '   ** ﾌﾟﾛｸﾞﾗﾑ ﾌｧｲﾙﾒｲ=ｼﾞｼｮｦﾂｶｳ     ﾄﾞﾗｲﾌﾞ=0
120 '
130 CLEAR 3000:DIM L$(25)
140 OPEN"I",#1,"ｼﾞｼｮ"
150 INPUT #1,N
160   DIM YOMI$(N)
170 FOR I=1 TO N
180   INPUT #1,YOMI$(I)
190 NEXT
200 CLOSE
210 'ｹﾝｻｸ
220 NUM=1:I=1:REM* NUM=ﾃﾞｰﾀﾉｶｽﾞ**I=ﾓｸﾃｷﾉﾓﾉﾉ ｶｽﾞ
230 INPUT"ﾖﾐｦ ﾄﾞｳｿﾞ";KENSAKU$
240 IF INSTR(YOMI$(NUM),KENSAKU$) THEN GOSUB 340
250 NUM=NUM+1:IF NUM>N THEN 270
260 GOTO 240
270 '
280 CLS
290 FOR J=1 TO I
300 PRINT@(J*18,0),VAL("&H"+L$(J))
310 NEXT
320 GOTO 320
330 'ﾖﾐﾉ ｵﾅｼﾞﾓﾉｦ ｱﾂﾒﾙ
340  L$(I)=RIGHT$(YOMI$(NUM),4):I=I+1
350 RETURN
360 REM *L$(I)=ﾖﾐｶﾞｵﾅｼﾞﾓﾉﾉ ｱﾂﾏﾘ
```

挨会藍合愛相哀

▲右のプログラムの出力例

**読みのデータから同音異義語をまとめる

ここでは，そういった問題はさておいて，いま作った読みのデータを使うことを考えます．上のサンプルを見てください．

まずファイルからデータを取出して，FM-77の内部メモリに入れてしまいます．そして行番号230で，探したい読みをまず求めます．ここではINSTR関数を使って，読みのデータの中から，探したいものを検索しています．そして，該当するものが見つかったら，もうひとつの配列にこのデータを入れるサブルーチンを呼出します．

ここでは L$(I) としていますが，この配列には，元のデータの中から，目的の読みに該当するものが集められ，そのコード部分がメモリされます．たとえば上の見本なら

L$(1)＝3027：L$(2)＝3271……

という形になります．

＊＊＊応用方法としては

いまの，いわば同音異義語を集めた配列は，とりあえず画面出力にだけ使われていますが，こうした処理を基本にした発展型としては
○プログラムに近い形で使い，出力された漢字に番号をつけ，番号入力によりコードを与えて文章化してゆく
○配列に番号をつけて，読みのキー（KEY：手がかり）にする．たとえば“アア＝1，アイ＝2，アウ＝3……”という形にして，入力された検索用の文字列から先頭の2文字を取出し，“アイ”なら2番といった番号づけのプログラムを作る．ファイルもあらかじめ1番のファイルは“アア”，2番は“アイ”というような作り方にして，目的の漢字をファイルから呼び出す．
といったことが考えられます．つまり配列にいったん納められた同音意義語もまた，外部記憶装置にデータの型で納める方法です．ランダムファイルを使えば，アクセス時間も速く，またFM-77のメモリも多くは使わないことになります．

＊＊＊＊読みの追加プログラムについて

さて，読みと，それに対応する漢字コードを探し出す方法についての2つの例を説明してきましたが，とりあえず2番目の，読みに対応する漢字コードを自分で作るプログラムをいったんまとめましょう．右には2つのプログラムが掲載されています．上のプログラムは，いわばメインプログラムで，メニューにしたがって3つのサブプログラムを呼出す形にしています．ここではCHAIN命令を使っていますが，1～3までを選ぶとそれぞれの仕事をしてくれます．

読みとそれに対応するコードの追加入力プログラムが，その下に出ています．シーケンシャルファイルを使って，データを追加する方法にはいくつかのものがありますが，ここではいちばん単純なものにしています．つまり，旧データが入っているファイルからデータを取出して，コンピュータの内部メモリに移したら，いったん旧ファイルは消してしまうのです．旧データからはいくつのデータがあったのか（ここでの変数はN）ということと，データそのもの（ここではYOMI＄）をもらって，再び新規入力と同じようにデータを入れてゆく方法になります．したがって，追加入力と新規入力（前のページ参照）はほとんど同じようなプログラムです．ひとつにまとめて，メニュー形式にして新規入力か，追加入力かを選ぶプログラムにするのが本筋ですが，ここでの目的は，読みと漢字の対応を考えることなので，個々のプロ

```
1 '=== ｼﾞｼｮﾆﾖﾙ ｶﾝｼﾞﾉ ｼｭﾂﾘｮｸ*MAIN PRO.====
2 WIDTH80,20:CLS
3 A$="ｼﾞｼｮｻｸｾｲ":B$="ｼﾞｼｮﾂｲｶ":C$="ｼﾞｼｮｦﾂｶｳ"
4 PRINT"1:";A$:PRINT"2:";B$:PRINT"3:";C$
5 INPUT"ﾄﾞﾚﾃﾞｽｶ";X:IF X<1 OR X>3 THEN 5
6 ON X GOTO 7,8,9
7 CHAIN A$:      REM SUB PRO.
8 CHAIN B$:      REM SUB PRO.
9 CHAIN C$:      REM SUB PRO.
```

メインプログラム▶

```
100 '========== ﾖﾐﾉ ｼﾞｼｮ ﾉ ﾂｲｶ SUB PRO. ==================='
110 '       * ﾌﾟﾛｸﾞﾗﾑ ﾌｧｲﾙﾒｲ=ｼﾞｼｮﾂｲｶ    ﾄﾞﾗｲﾌﾞ=0
120 CLEAR 3000:DEFINT A-Z:DIM YOMI$(2000)
130 OPEN"I",#1,"ｼﾞｼｮ"
140   INPUT #1,N
150   FOR I=1 TO N
160   INPUT #1,YOMI$(I)
170 NEXT
180 CLOSE
190 KILL "ｼﾞｼｮ" :'ﾃﾞｰﾀｦ ﾊｲﾚﾂﾆｲﾚﾀﾉﾃﾞ ﾌﾙｲﾓﾉｦ ｹｽ
200 'ﾂｲｶ
210 N=N+1
220 CLS
230 LOCATE 0,5:INPUT"ﾖﾐ  ";YOMI$
240   INPUT"ｺｰﾄﾞ ";KCODE$:PRINT@(50,10),VAL("&H"+KCODE$)
250   YOMI$(N)=YOMI$+KCODE$
260   PRINT"ﾃｲｾｲ ･･･>T  ｵﾜﾘ････>E  ﾂｷﾞﾉﾆｭｳﾘｮｸ  ANY KEY"
270    YN$=INKEY$:IF YN$="" THEN 270
280 IF YN$="T" OR YN$="t" THEN 220
290 IF YN$="E" OR YN$="e" THEN 320
300 N=N+1:IF N>2000 THEN 320 ELSE GOTO 220
310 'ﾌｧｲﾙ ｻｸｾｲ
320 OPEN"O",#1,"ｼﾞｼｮ"
330 PRINT #1,N
340 FOR I=1 TO N
350  PRINT #1,YOMI$(I)
360 NEXT
370 CLOSE
```

“読み”の追加プログラム▶

グラムにして理解しやすいように処理しています．

というわけで，ひとまず漢字の話を終えましょう．なお，カタカナやひらがなも漢字の仲間として扱わないと文章になりませんので，つぎのページでは漢字以外の文字と，その探しかたについてを考えてゆきましょう．

ひらがな，カタカナの扱いかた

＊カナにも専用の配列を用意する

かなは読みかたが固定しているので，その処理はむずかしくはありません．ただ，注意しないといけないのはFM-77のキーボードにはないカタカナやひらがながあることです（ゐ，ゑ，ヰ，ヱ，ヴなど）．これらは今日では死語に等しいので，とりあえずは，ゐ…イ，ゑ…エなどと同じにしてもよいでしょう．

右のページのサンプルは今まで説明してきた手法をもとにしています．まずサブルーチンの310行以降で，KANA＄（I）に

ぁ　2421　　あ　2422　　ぃ　2423

といった文字列を作り，それをしまっておきます．そしてキーボードから入力した文字と，この配列にしまわれている文字を比較して，目的のものを見つけ出すわけです．

＊＊データを２分して検索をさせる方法

JISのひらがなは，83個あります．キーボードから入力したカナと，配列にしまわれているデータを照合するとき，入力文字に関係なくデータの先頭から探しはじめると，“ン”がもっとも時間がかかります．もっとも，FM-77は処理速度が速いので，100個程度のデータから目的のものを見つけるとしても，実際には0.5秒もかからないでしょう．しかし，ここではより速く探し出すアルゴリズムの例を考えてみます．

行番号250を見てください．入力されたカナが“ト”よりも小さい…つまりASCIIのコード表で「ヲ，ァ，…，テ，ト」までの文字であったら，検索開始は１番目から，“ト”よりも大きかったら検索開始は42番目からという意味の数値をKSという変数に与えています．

たとえば入力された文字が“ナ”だとしたら，KS＝41になり，行番号250のインストリング関数は，まず

IF INSTR（KANA$（41），ナ） THEN

になります．KANA＄（41）は“ド”ですから，この判断文は偽になり，KS＝KS＋1…42 となって，また探し始めます．そして，２回目はKANA＄（42）になり，ここに“ナ”があるので，行番号280に飛び，“ナ”の文字列の右から４つのつづりをカナのコードとして取出してくれます．順序づけがハッキリしているデータなら，このように開始位置が指定でき，すばやい検索が可能です．

```
100 '==================== ｶﾀｶﾅ ﾋﾗｶﾞﾅﾉ ｹﾝｻｸ ﾄ ｶﾞﾒﾝ ｼｭﾂﾘｮｸ =======================
110 DEFINT A-Z:CLEAR 1000:WIDTH80,25:COLOR 7:CLS
120 '
130 GOSUB 310:'.........................................ｶﾅｺｰﾄﾞ ﾃｲｷﾞ SUB
140 INPUT"1:ﾋﾗｶﾞﾅ  2:ｶﾀｶﾅ  ﾄﾞﾁﾗ";KH:IF KH=2 THEN KH=256 ELSE KH=0
150 '-------------------- M A I N   R O U T I N E --------------------------
160 CLS
170 LOCATE 0,10:INPUT"ﾓｼﾞｦ ﾄﾞｳｿﾞ";X$:GOSUB 240:'....................ﾍﾝｶﾝ SUB
180      PRINT@(M*20,20),(KANA+KH);:REM KH=ｶﾀｶﾅ ﾋﾗｶﾞﾅ  ｸﾍﾞﾂｮｳ
190      LOCATE 0,12:PRINT"ﾂｷﾞｦ ﾀｼｶﾒﾙﾅﾗ ANY KEY PUSH"
200 LOCATE 0,10:PRINT"                    ":M=M+1:GOTO 170
210 '--------------------------------------------------------------------------
220 '
230 'ｶﾅﾆｭｳﾘｮｸﾆ ﾀｲｵｳｽﾙ ｺｰﾄﾞｦ ｻｶﾞｽ
240 IF X$<="ﾄ"THEN KS=1ELSE KS=41:'............................ｹﾝｻｸ ｶｲｼｲﾁｦ ｷﾒﾙ
250     IF INSTR(KANA$(KS),X$) THEN 280 ELSE KS=KS+1:GOTO 250
260 '
270 '4ｹﾀﾉ ﾓｼﾞｦ ﾄﾘﾀﾞｼ,ｶﾅﾉ ｺｰﾄﾞﾆﾍﾝｶﾝ
280 KANA$=RIGHT$(KANA$(KS),4):KANA=VAL("&H"+KANA$):RETURN
290 '==========================================================================
300 'ｶﾀｶﾅ ﾋﾗｶﾞﾅｦ ｶﾝｼﾞCODEﾆｽﾙﾀﾒﾉ DATA ﾆｭｳﾘｮｸ
310 DIM KANA$(86):HIRA=&H1:CODE=&H2420:RESTORE 380 :'9248=(&H2420)
320 WHILE KANA$<>"THEEND"
330   READ KANA$:KANA$(HIRA)=KANA$+HEX$(CODE+HIRA):HIRA=HIRA+&H1
340 WEND
350 RETURN
360 '
370 '**** ｶﾅ ﾃﾞｰﾀ ****
380 DATA ｧ,ｱ,ｨ,ｲ,ｩ,ｳ,ｪ,ｴ,ｫ,ｵ,        ｶ,ｶﾞ,ｷ,ｷﾞ,ｸ,ｸﾞ,ｹ,ｹﾞ,ｺ,ｺﾞ
390 DATA ｻ,ｻﾞ,ｼ,ｼﾞ,ｽ,ｽﾞ,ｾ,ｾﾞ,ｿ,ｿﾞ,   ﾀ,ﾀﾞ,ﾁ,ﾁﾞ,ｯ,ﾂ,ﾂﾞ,ﾃ,ﾃﾞ,ﾄ
400 DATA ﾄﾞ,ﾅ,ﾆ,ﾇ,ﾈ,ﾉ,ﾊ,ﾊﾞ,ﾊﾟ,ﾋ,     ﾋﾞ,ﾋﾟ,ﾌ,ﾌﾞ,ﾌﾟ,ﾍ,ﾍﾞ,ﾍﾟ,ﾎ,ﾎﾞ
410 DATA ﾎﾟ,ﾏ,ﾐ,ﾑ,ﾒ,ﾓ,ｬ,ﾔ,ｭ,ﾕ,      ｮ,ﾖ,ﾗ,ﾘ,ﾙ,ﾚ,ﾛ,ヮ,ﾜ,ｲ,          ｴ,ｦ,ﾝ
420 DATA "THEEND"
```

というわけで，検索時間は半分に短縮されるわけですが，実際には前にも述べたように，JISの一覧表とASCIIのコード体系は異なるため，キーボードからの入力文字が“ョ”のとき，このプログラムでは“ト”よりも小さいと判断をして，1番目から探し始めます．しかしJISの体系ではずっとうしろの方にあるので，本当はKS＝42から検索させるべきなのです．特定条件をつけた文を行番号250に追加しておけばよい（ホ，ャ，ュ，ョが入力されたら，KS＝42とする）のですが，条件文を書くとそれを判断するための時間がかかります．したがって実用上はどちらが本当に速いかを考えないといけませんが，ここでは無視しています．

文書を作り，保存する

＊画面の基本設計をどうするか

では，以上の予備知識をふまえて，画面に文章を作りひとつの文書としてまとめたのち，フロッピーに記録するプログラムを作ることにしましょう．ワードプロセッサには程遠いとしても，郵便物の宛名書き程度には使えます．まず考えることは，1文書を，どのようなスタイルにまとめるのか，です．ここではもっともやさしく作れるものとして1行あたり40桁（40文字），6行までの文書とします．FM-77のグラフィック画面は640×200ドットですから，漢字（1字あたり16×16ドット）は1行に40文字，そして12行まで（16×12＝192ドット）出力できるのですが，行と行（上下関係）は少し間隔を空けないと，とても見にくくなります．また，メニューのスペースを作らないと操作できないので，ここでは6行に抑え，その代り画面の上下に余裕をもたせることにしました．

＊＊用意すべきおもな変数について

まず用意しなくてはいけない変数としては○漢字検索用配列　○カナ検索用配列　○作成した文章記録用配列　があります．文章のデータは文字列変数にしてもよいのですが，メモリの効率からいえば数値変数にするべきです．ここではBUN（5, 40）にしていますが，この配列に実際に代入されるのは，たとえば0行目の左から5番目の文字データならBUN（0, 4）になり，もしこの位置に“山”という字が入っていれば，“山”のコードは&H3B33なのでBUN（0, 4）＝&H3B33（10進数の15155）になります．このプログラムで扱う数値は整数だけなのですべての変数に整数型を使うことを宣言しておきます（行番号170）．

＊＊＊重要な，文字数と行数の管理

さて，このプログラムでは，行数は0～5までの6行，文字数はおなじく0～39までの40文字ですから，画面出力および配列変数のBUN（5, 40）をコントロールするためには，常に行と文字数をチェックし，正しくカウントします．たとえば

○文字数が39を越えたら，つぎの行になるので，行に1を加える

○ある行の文字数が39になる前に文章を改行したら，新しい行には1を加え，文字数は0にもどしてやる

○出力した文字を訂正するときは，同じ行の中なら文字数をひとつ減

```
100 '========================== ｶﾝｼﾞﾉ ﾌﾞﾝｼｮ ===================================
110 '
120 '                     * 1ｷﾞｮｳ 40 ﾓｼﾞ 6ｷﾞｮｳﾌﾞﾝﾉ ﾌﾞﾝｼｮ ﾎｿﾞﾝ *
130 '
140 '                                                          JUN '84  by T.U
150 '=========================================================================
160 '
170 DEFINT A-Z:CLEAR 3000:WIDTH80,25:COLOR 7:CLS
180 DIM BUN(5,40) :'......................................ﾌﾞﾝｼｮDATAﾖｳ ﾊｲﾚﾂ
190 ' ｶﾝｼﾞｺｰﾄﾞｦ ｹﾝｻｸｽﾙ ﾀﾒﾉ READ DATA
200 DIM YOMI$(47):KAN=1:RESTORE 1440
210    WHILE YOMI$<>"THEEND"
220       READ YOMI$:YOMI$(KAN)=YOMI$:KAN=KAN+1
230    WEND
240 '
250 'ｶﾀｶﾅ ﾋﾗｶﾞﾅ ｺｰﾄﾞｦ ｹﾝｻｸｽﾙﾀﾒﾉ READ DATA
260 DIM KANA$(84):HIRA=1:CODE=9248:RESTORE 1520 :'..................9248=&H2420
270    WHILE KANA$<>"THEEND"
280        READ KANA$:KANA$(HIRA)=KANA$+STR$(CODE+HIRA):HIRA=HIRA+1
290    WEND
300 '
310 '---------------------- ﾆｭｳﾘｮｸ ﾉ ｾﾂﾒｲｦ ｶﾞﾒﾝﾆ ﾀﾞｽ -------------------------
320 FOR I=0 TO 38 STEP 2:LOCATE I*2,0:PRINTI;:NEXT
330 FOR I=0 TO 9:SYMBOL(25*I,145),STR$(I),1,1,5:NEXT
340 LOCATE 40,23:PRINT"1ｼﾞｵｸﾘ=SPACE  1ｼﾞﾓﾄﾞﾘ=BS  ｶｲｷﾞｮｳ=RET";
350 LOCATE 0,24:COLOR 5:PRINT"0:DIRECT  1:ﾋﾗ   2:ｶﾀ   3:ｴｲｽｳ   4:ｷｺﾞｳ   5:ｶﾅ-->ｶ
ﾝｼﾞ  E:END";:COLOR7
360 '
370 '============================ ﾒｲﾝ ﾙｰﾁﾝ ===================================
380 REM *GYO=ｷﾞｮｳｽｳ  *MOJI=1ｷﾞｮｳﾉ ﾓｼﾞｽｳ  *KANJI (&H3020=" "・・・・・ｶﾗﾓｼﾞ)
390    GYO=0:MOJI=0
400 COLOR 5:PRINT@(MOJI*16,GYO*22+10),&H2132:COLOR 7:'....&H2132=("_" ｶｰｿﾙｶﾞﾜﾘ)
410 LOCATE 0,22:PRINT"ﾅﾆｦ ｾﾝﾀｸ ｼﾏｽｶ(0・・・8)? ";
420     S$=INKEY$:IF S$="" THEN 420
430 IF S$=CHR$(&H0D) THEN GOSUB 590:GYO=GYO+1:MOJI=0:IF GYO=6 THEN 660 ELSE 400
440 IF S$=CHR$(&H20) THEN GOSUB 590:KANJI=&H3020:GOTO 510 :'.............SPACE
450 IF S$=CHR$(&H08) THEN GOSUB 610:GOTO 400 :'...........................BS
460 IF S$="E" THEN 660 :'...................................SAVE ﾙｰﾁﾝ
470          S=VAL(S$):IF S<0 OR S>5 THEN BEEP:GOSUB 570:GOTO 410
480 IF S=0 THEN GOSUB 850:GOTO 500 :'......................CODEﾆｭｳﾘｮｸ
490       ON S GOSUB 920,1020,1110,1190,1200
500 GOSUB 590 :'...........................................ｶｰｿﾙｦ ｹｽ
510    PRINT@(MOJI*16,GYO*22+10),KANJI:BUN(GYO,MOJI)=KANJI
520    MOJI=MOJI+1:IF MOJI MOD 39=0 THEN GYO=GYO+1:MOJI=0
530 IF GYO>=6 THEN 660 :'...................................SAVE ﾙｰﾁﾝ
540 GOSUB 570:GOTO 400 :'...................................ﾂｷﾞﾉ ﾆｭｳﾘｮｸ
550 '
```

らすので，文字をカウントする変数もひとつ減らす
○行数のカウントが5になり，しかも文字数も39になったら，それ以上の文章は作成できないように処理をし，フロッピーに記録するようなプログラムにする.

******メインルーチンはこうなっている**

まずメインルーチンを見てください．行番号400で，現在のカーソルを画面に出力します．行（変数GYO），文字（変数MOJI）とも0ですから，最初は当然左上にカーソルが出ます．文字の出力位置は行と文字の変数を使います．行番号430は改行（操作キーはリターンキー）440が1字送り（操作はスペースバー），450は1字戻し（操作はバックスペースキー）になります.

つぎに，入力モードを受付けます．ここでは0が直接入力（出力したい文字をコードで入力する），1がひらがな，2がカタカナ……となっているので，行番号490で入力モードに合せたサブルーチンを呼出すようにしています．それぞれのサブルーチンで，使い手が入力した文字に合致するコードを探したら，行番号510でそれを画面に出します．画面に出したあと，行番号520で，文字数をひとつ増やすと共に，もし文字数が39になったら（ここではMODを使って判断），行数はひとつ増やし，文字数は0にもどしています.

この操作のくりかえしで，設定文字数および行数になるまで文章を作り，作業を終えたら記録ルーチンに飛んで，画面に出力されたデータをファイルにしまうわけです.

*******その他の注意点について**

なお，使用頻度の高い句読点(くとう)（。や，）などは，文字に対応するコードをいちいち検索しなくても入力できるように，ここでは英数，ひらがな，カタカナの3つのモードの時にはすぐに画面に出力するようにしました．横書きの文章ですから，カンマだけにして，日本語の読点(、)は使わないようなプログラムです（本書でも，カンマとピリオドを使っています．本によっては句点＝「。」と読点＝「、」を使うものもあります）.

また，1字分を戻すときは特別な注意が必要です．というのは，1字分戻した時には画面に出た漢字を消さないといけないのですが，漢字には空白の文字がないからです．行番号620のように，LINE文とボックスフィル（BF）を使って消すしか方法はありません.

```
560 '**** ｶﾞﾒﾝｹｼ SUB
570       FOR E=19 TO  23:LOCATE 0,E:PRINT SPACE$(39);:NEXT:RETURN
580 '**** ｶｰｿﾙｹｼ SUB
590       COLOR 0:PRINT@(MOJI*16,GYO*22+10),&H2132:COLOR7:RETURN
600 '**** BACKSPACESUB
610     GOSUB 590:MOJI=MOJI-1:IF MOJI<0 THEN MOJI=0:'···············ﾓｼﾞｽｳｦ ﾍﾗｽ
620     LINE(MOJI*16,GYO*22+10)-(MOJI*16+16,GYO*22+26),PSET,0,BF:'········ﾓｼﾞｦｹｽ
630 RETURN
640 '
650 '----------------------- ｼｰｹﾝｼｬﾙ ﾌｧｲﾙﾍﾉ ｷﾛｸ-----------------------------------
660 GOSUB 570:'·······················································ｶﾞﾒﾝｹｼ
670   BEEP:LOCATE 0,22:PRINT"ﾌｧｲﾙﾆ ｷﾛｸｼﾏｽｶ? Y or N (ﾄﾞﾗｲﾌﾞ=0)";
680 YN$=INKEY$:IF INKEY$="" THEN 680
690   IF YN$="N" OR YN$="n" THEN CLS:GOTO 320
700 IF YN$="y" OR YN$="y" THEN 710 ELSE GOTO 680
710   GOSUB 570:LOCATE 0,22:INPUT"ﾌﾞﾝｼｮﾒｲ(5ﾓｼﾞ)? ";BUN$
720   BUN$=LEFT$(BUN$,5)
730 OPEN"O",#1,"0:ｶﾝ."+BUN$
740   FOR G=0 TO 5:FOR M=0 TO 39:PRINT #1,BUN(G,M):NEXT M,G:CLOSE
750 ERASE BUN:DIM BUN(5,40):CLS:GOTO 320
760 '
770 '**** ｺﾝﾏ ﾋﾟﾘｵﾄﾞ etc. ﾆｭｳﾘｮｸ SUB
780     IF HIRA$="," THEN KANJI=&H2124:RETURN500
790     IF HIRA$="." THEN KANJI=&H2125:RETURN500
800     IF HIRA$="｡" THEN KANJI=&H2123:RETURN500
810     IF HIRA$="-" THEN KANJI=&H213C:RETURN500
820 RETURN
830 '
840 '----------------------- ﾁｮｸｾﾂ ﾆｭｳﾘｮｸ -----------------------------------
850 GOSUB 570:LOCATE 0,22:PRINT"CODE(16ｼﾝ) ｦ ﾄﾞｳｿﾞ";
860 KANJI$=INPUT$(4):KANJI=VAL("&H"+KANJI$) :'·······················CODEﾍﾝｶﾝ
870   PRINT@(10,155),KANJI:LOCATE 0,23:PRINT"OK? Y or N";
880   YN$=INKEY$:IF YN$="" THEN 880
890   IF YN$="N" OR YN$="n" THEN GOSUB 570:GOTO 400
900   IF YN$="Y" OR YN$="y" THEN RETURN
910 BEEP:GOTO 880
920 '------------------------------- ﾋﾗｶﾞﾅ ------------------------------------
930 '
940 GOSUB 570
950   LOCATE 0,22:LINEINPUT"ﾓｼﾞｦ ﾄﾞｳｿﾞ  ";HIRA$
960 GOSUB 780 :'············································ｺﾝﾏ ﾋﾟﾘｵﾄﾞSUB
970 '** ｹﾝｻｸ
980 I=1
990   IF INSTR(KANA$(I),HIRA$) THEN KANJI=VAL(RIGHT$(KANA$(I),4)):RETURN
1000   I=I+1:IF I<=HIRA THEN 990 ELSE RETURN 400
1010 GOTO 990
1020 '------------------------------- ｶﾀｶﾅ -----------------------------------
```

```
1030 GOSUB 570
1040    LOCATE 0,22:LINEINPUT"ﾓｼﾞｦ ﾄﾞｳｿﾞ  ";HIRA$
1050 GOSUB 780 :'········································ｺﾝﾏ ﾋﾟﾘｵﾄﾞ SUB
1060 '** ｹﾝｻｸ
1070 I=1
1080  IF INSTR(KANA$(I),HIRA$) THEN KANJI=256+VAL(RIGHT$(KANA$(I),4)):RETURN
1090  I=I+1:IF I<=HIRA THEN 1080 ELSE RETURN 400
1100 GOTO 1080
1110 '---------------------------------ｴｲｽｳ ----------------------------------
1120 GOSUB 570:LOCATE 0,22:PRINT"ﾓｼﾞｦ ﾄﾞｳｿﾞ ";
1130    EISU$=INPUT$(1):PRINT EISU$;"            ";
1140 HIRA$=EISU$:GOSUB 780:EISU$=HIRA$ :'·············ｺﾝﾏ ﾋﾟﾘｵﾄﾞ SUBﾍﾉ ﾍﾝｽｳﾋｷﾜﾀｼ
1150   IF ASC(EISU$)>=48 AND ASC(EISU$)<=57 THEN KANJI=(9008-48)+ASC(EISU$):RETU
RN:'ｽｳｼﾞ
1160   IF ASC(EISU$)>=65 AND ASC(EISU$)<=90 THEN KANJI=(9025-65)+ASC(EISU$):RETU
RN:'ALPHA ｵｵﾓｼﾞ
1170   IF ASC(EISU$)>=97 AND ASC(EISU$)<=122THEN KANJI=(9057-97)+ASC(EISU$):RETU
RN:'ALPHA ｺﾓｼﾞ
1180 '---------------------------- ｷｺﾞｳ ------------------------------------
1190 YS=&H21E0:GOSUB 1320:RETURN
1200 '-----------------------  ｶﾅ ----> ｶﾝｼﾞ ｻｰﾁ  -------------------------
1210 GOSUB 570:LOCATE 0,22:PRINT"ﾖﾐﾉ ﾊｼﾞﾒﾉ ﾓｼﾞ ? ";
1220   Y$=INPUT$(1):PRINT Y$;
1230   IF ASC(Y$)<&HB1 OR ASC(Y$)>&HDC THEN BEEP:RETURN 400
1240 '** ｹﾝｻｸ
1250 I=1
1260   IF Y$=LEFT$(YOMI$(I),1) THEN YS$=RIGHT$(YOMI$(I),4):GOTO 1280
1270   I=I+1:IF I<=KAN THEN GOTO 1260 ELSE RETURN 400
1280 YS=VAL("&H"+YS$):GOSUB 1320:RETURN
1290 '
1300 '****** ﾋｮｳｼﾞ･ｾﾝﾀｸ SUB ***********
1310 REM ** SPASE=ｶﾝｼﾞﾄ ｶﾝｼﾞﾉ ｶﾝｶｸｦ ｱｹﾙﾀﾒﾉ ﾍﾝｽｳ
1320   FOR J=0 TO 9:PRINT @(J*26,155),YS+J:NEXT
1330   LOCATE 0,22:PRINT"ﾅﾝﾊﾞﾝﾒ? ｼﾞｺｳﾎ <--  --> ,ﾄﾘﾔﾒ=ESC";
1340   K$=INKEY$:IF K$="" THEN 1340
1350   IF ASC(K$)=&H1C THEN YS=YS+10:GOSUB 570:GOTO 1320 :     '&H1C="--->"
1360   IF ASC(K$)=&H1D THEN YS=YS-10:GOSUB 570:GOTO 1320 :     '&H1D="<---"
1370   IF ASC(K$)=&H1B THEN GOSUB 570:RETURN 400 :             '&H1B=ESC KEY
1380   IF ASC(K$)<&H30 OR ASC(K$)>&H39 THEN BEEP:GOTO 1330:    'ｽｳｼﾞﾉﾐ ｳｹﾂｹﾙ
1390   KANJI=VAL(K$):IF  KANJI<0 OR KANJI>9 THEN 1330 ELSE KANJI=KANJI+YS
1400 RETURN
1410 '
1420 '
1430 '**** ｶﾝｼﾞ ﾃﾞｰﾀ ****
1440 DATA ｱ3036,ｲ3065,ｳ3133,ｴ3156,ｵ3226,ｶ3346,ｷ3563,ｸ3675,ｹ3767,ｺ3929
1450 DATA ｻ3A59,ｼ3C3A,ｽ3F65,ｾ4067,ｿ4168,ﾀ4265,ﾁ4365,ﾂ4454,ﾃ4478,ﾄ4576
1460 DATA ﾅ4668,ﾆ467A,ﾇ4728,ﾈ4729,ﾉ4735,ﾊ4772,ﾋ487E,ﾌ4975,ﾍ4A44,ﾎ4B21
```

```
1470 DATA ﾏ4B6D,ﾐ4C2C,ﾑ4C33,ﾒ4C44,ﾓ4C53,ﾔ4C70,ﾕ4D27,ﾖ4D51,ﾗ4D6B,ﾘ4E35
1480 DATA ﾙ4E5C,ﾚ4E73,ﾛ4F34,ﾜ4F4D
1490 DATA "THEEND"
1500 '
1510 '**** ｶﾅ ﾃﾞｰﾀ ****
1520 DATA ｧ,ｱ,ｨ,ｲ,ｩ,ｳ,ｪ,ｴ,ｫ,ｵ,ｶ,ｶﾞ,ｷ,ｷﾞ,ｸ,ｸﾞ,ｹ,ｹﾞ,ｺ,ｺﾞ,ｻ,ｻﾞ,ｼ,ｼﾞ,ｽ,ｽﾞ,ｾ,ｾﾞ
1530 DATA ｿ,ｿﾞ,ﾀ,ﾀﾞ,ﾁ,ﾁﾞ,ｯ,ﾂ,ﾂﾞ,ﾃ,ﾃﾞ,ﾄ,ﾄﾞ,ﾅ,ﾆ,ﾇ,ﾈ,ﾉ,ﾊ,ﾊﾞ,ﾊﾟ,ﾋ,ﾋﾞ,ﾋﾟ,ﾌ,ﾌﾞ,ﾌﾟ
1540 DATA ﾍ,ﾍﾞ,ﾍﾟ,ﾎ,ﾎﾞ,ﾎﾟ,ﾏ,ﾐ,ﾑ,ﾒ,ﾓ,ｬ,ﾔ,ｭ,ﾕ,ｮ,ﾖ,ﾗ,ﾘ,ﾙ,ﾚ,ﾛ,*,ﾜ,ｲ,ｴ,ｦ,ﾝ
1550 DATA "THEEND"
```

招　待　状

つぎの日曜日に，我家において盛大なるパーティを開催することになりました。
つきましては，当日の午後5時までにおこしください。
なお，ささやかですが晩餐の用意がございますので，あらかじめお知らせいたします

12月20日

上柿 力　拝

▲出力のサンプル

```
100 '============== ｶﾝｼﾞ ﾃﾞｰﾀｦ ﾖﾋﾞﾀﾞｽ ===================
110 DEFINT A-Z:WIDTH 80,25
120 DIM BUN(5,40)
130 INPUT"ﾌﾞﾝｼｮﾒｲ(5ﾓｼﾞ)";BUN$
140  BUN$=LEFT$(BUN$,5)
150 '
160 OPEN"I",#1,"0:ｶﾝ."+BUN$
170 FOR G=0 TO 5:FOR M=0 TO 39:INPUT #1,BUN(G,M):NEXT M,G
180 CLOSE
190 '
200 CLS
210 FOR G=0 TO 5:FOR M=0 TO 39
220 GYO=G*22:'.........................ｷﾞｮｳｶﾝｶｸｦ ｱｹﾙ
230 PRINT@(M*16,GYO),BUN(G,M):'........ｶﾝｼﾞ ｼｭﾂﾘｮｸ
240 NEXT M,G
250 '
260 LOCATE 0,22:PRINT"ﾊｰﾄﾞCOPY ｼﾏｽｶ?"
270 YN$=INKEY$:IF YN$="" THEN 270
280 IF YN$="Y" OR YN$="y" THEN HARDC2:END
290 IF YN$="N" OR YN$="n" THEN END ELSE 270
```

文書を呼び出し，出力する▶プログラム

ここに紹介したプログラムを使って，画面に文書だけを出力するのが上のプログラムです．データファイルを呼び出して，ハードコピー命令を使って印字させています．

漢字フォントを拡大して見る

＊GET@を応用して拡大する

漢字は16×16ドットで構成されていますが，画数の多い文字などは正確には表わせず，それらしく見えるように適度に処理をしています．字体（フォント）がどうなっているのかを，漢字を拡大して出力してみましょう．

ここではGET@の形式３を使い，１ドットごとにその有無を調べています．サブルーチンはこのドットをもとに，拡大率に従った幅で別の場所にPUT@してゆきます．たとえばX方向の拡大率が10倍のとき，XXW＝9になります．いま，入力をした文字が（X＝0）の位置に点灯していると，サブルーチンでは

XW＝1からXXWまで（X＝1，2，3，4，……，9）

XX＝0×XMAG＝0×10＝0

X0＝250なので，PUT@されるX座標は251～259

同様にX＝1の点はX方向に261～269というように拡大されます（Y方向もおなじ）．

なお，拡大率が低いとPUT@する点が少なく，見にくくなりますから10×5（X×Y）倍程度にするとよいでしょう．

```
100 '************************* ｶﾝｼﾞﾉ ﾊﾟﾀｰﾝ ｦ ﾐﾙ ******************************
110 '
120              DEFINT A-Z:WIDTH 80,25
130 INPUT"ﾖｺﾉ ｶｸﾀﾞｲﾘﾂ(2ﾊﾞｲ ｲｼﾞｮｳ) ";XMAG:INPUT"ﾀﾃﾉ ｶｸﾀﾞｲﾘﾂ(2ﾊﾞｲ ｲｼﾞｮｳ) ";YMAG
140     YYW=YMAG-1:XXW=XMAG-1:X0=250:Y0=20:DIM KAN(3)
150 INPUT"ﾓｼﾞﾊ ";CODE$:K=VAL("&H"+CODE$)
160     CLS:LOCATE 5,1:PRINT "ｶﾝｼﾞCODE=&H";CODE$;:PRINT@(0,0),K
170  FOR Y=0 10 15:FOR X=0 TO 15
180    GET@(X,Y)-(X,Y),KAN,G,7:GOSUB 230
190  NEXT X:NEXT Y
200 LOCATE 30,24:PRINT"ﾂｷﾞｦ ﾐﾙﾅﾗ ANY KEY PUSH";
210        WHILE INKEY$="":WEND:CLS:GOTO 150
220 '
230 FOR XW=1 TO XXW:FOR YW=1 TO YYW
240     YY=Y*YMAG:XX=X*XMAG
250     PUT@(X0+XX+XW,Y0+YY+YW)-(X0+XX+XW,Y0+YY+YW),KAN,PSET,6
260 NEXT YW:NEXT XW
270 RETURN
```

7
サウンド&ミュージック

サウンドレジスタの基本

＊発振音の高さをきめる

※周波数 (frequency)： 1秒間にくりかえされる物体や電気の振動数．1秒間に100回のくりかえしは100Hz (ヘルツ)，1秒間に10,000回のくりかえしは10,000Hzというように表示する

※CT, FTと発振周波数

50Hz	CT=6	FT=1
100Hz	CT=3	FT=1
300Hz	CT=1	FT=1
500Hz	CT=0	FT=153
1,000Hz	CT=0	FT=77
2,000Hz	CT=0	FT=38
4,000Hz	CT=0	FT=19
6,000Hz	CT=0	FT=13

FM-77に内蔵されているサウンドICは，基本的には3つの音を独立して発生する仕様になっています．さらにノイズ発生器・波形加工器をそれぞれ1個ずつ持ったもので，その構成はマニュアルに掲載されているように，レジスタ（register：データを一時的に置く装置）の7番を中心にして，2つのブロックに分れています．

● 方形波およびノイズ発生ブロック（レジスタ0～6）
● 音量および波形加工ブロック（レジスタ8～13）

まず方形波とノイズ発生レジスタを見ましょう．方形波の発生は3つのチャネルがありレジスタ0・1，2・3，4・5はまったくおなじ使いかたになります．それぞれのレジスタにデータを入れると，そのデータに従った音を発生しますが，発振周波数は

$f=(76800)\div(256\times CT+FT)$ (Hz)

CT＝レジスタ0, 2, 4に与えるデータ（数値は0～15）

FT＝レジスタ1, 3, 5に与えるデータ（数値は0～255）

になります．サンプルプログラムはCTとFTを与え，その音の高さ（周波数）を出力すると共に，音と波形をディスプレイするものです．

＊＊ノイズと波形加工用の発振周波数

※ノイズ (noise)：一定のくりかえしもせず，いろいろな音が混在しているもの．FM-77ではホワイトノイズと呼ばれる，あらゆる周波数が混在したものを発生する

※NTとノイズ発振周波数

4,800Hz	NT=1
2,400Hz	NT=2
1,200Hz	NT=4
800Hz	NT=6
150Hz	NT=31

いっぽうレジスタ6に割当てられているノイズ発振用レジスタに入れるデータはひとつで，NTは0～31の5ビットとなっています．

$f=(76800)\div(16\times NT)$ (Hz)

以上がレジスタ7を経由する発振器ですが，もうひとつ，波形加工用の発振器があります．こちらは

$f=(76800)\div[(256\times CT+FT)\times16]$ (Hz)

CT＝レジスタ12に与えるデータ（数値は0～255）

FT＝レジスタ11に与えるデータ（数値は0～255）

となっています．波形加工用の発振器は，レジスタ7を経由した音を加工する，きわめて低い周波数で，ふつうは1Hz～20Hz程度に設定します．

なお，これらのレジスタへの書きかたはすべて

SOUND〔レジスタ番号〕,〔数値〕

という形になります．

```
100 '================= sound frequency <<REGISTER 0 & 1>> =====================
110 TW=5000:WIDTH 80,25:CONSOLE 15,10:CLS:'・・・・・・・・・・・・・・・・・・TW=ｼﾞｶﾝｼﾞｸ ｹｲｻﾝﾖｳ
120 INPUT"CT(0-15)  ";CT%:IF CT%>15 THEN BEEP:GOTO 120
130 INPUT"FT(0-255) ";FT%:IF FT%>255 THEN BEEP:GOTO 130
140   IF CT%=0 AND FT%=0 THEN BEEP:GOTO 120
150 GOSUB 200:GOSUB 250:PRINT"ﾂﾂﾞｹ ﾏｽｶ PUSH Y or N "
170   YN$=INPUT$(1):IF YN$="Y" OR YN$="y" THEN SOUND 8,0:CLS:GOTO 120
180   IF YN$="N" OR YN$="n" THEN SOUND 8,0:WIDTH 80,25:CLS:END
190 GOTO 170
200 '---------------------- ﾊｯｼﾝ ｼｭｳﾊｽｳﾉ ｹｲｻﾝSUB--------------------------
210    F=76800!/(256*CT%+FT%)
220 PRINT USING"f=#######,.##Hz";F
230   SOUND 8,10:SOUND 0,FT%:SOUND 1,CT%
240 RETURN
250 '------------------------- ﾃﾞｨｽﾌﾟﾚｲSUB-----------------------------------
260 LINE(10,10)-(10,90),PSET,6:LINE(10,50)-(620,50),PSET,6:'・・・・・・・・・ﾀﾃ・ﾖｺｼﾞｸ
270 LINE(10+.05*TW,45)-(10+.05*TW,55),PSET,6:LINE(10+.1*TW,45)-(10+.1*TW,55),PSE
T,6:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾒﾓﾘ
280 IF F<=100 THEN MAG=10 ELSE IF F>100 AND F<=1000 THEN MAG=100 ELSE IF F>1000
AND F<=10000 THEN MAG=1000 ELSE IF F>10000 THEN MAG=10000:'・・・・・・・・ｼﾞｶﾝｼﾞｸﾁｮｳｾｲ
290 MAG$=STR$(MAG):MAG$=RIGHT$(MAG$,LEN(MAG$)-1):DISP$="1/"+MAG$+"sec"
300 HL=INT(.1*MAG*TW/(F*2)):S=10:'・・・・・・・・・・・・・・・・・・・・・・・・ﾊｹｲﾉ ﾊﾊﾞ:S ﾊ ｽﾀｰﾄｲﾁ
310 SYMBOL(500,55),DISP$,1,1,6:SYMBOL(0,46),"0",1,1,6:SYMBOL(5,0),"+",1,1,6:SYMB
OL(5,92),"-",1,1,6:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾓｼﾞｦ ｼｭﾂﾘｮｸ
320 CONNECT(S,20)-(S+HL,20)-(S+HL,80)-(S+2*HL,80)-(S+2*HL,20),4:'・・・・ﾊｹｲｦ ｴｶﾞｸ
330    S=S+2*HL:IF S>600 THEN 350
340 GOTO 320
350     LINE(610,20)-(639,80),PSET,0,BF:'ﾐｷﾞﾉ ﾌﾋﾂﾖｳﾌﾞﾌﾞﾝｦ ｹｽ
360 RETURN
```

***レジスタ7の使いかたについて

レジスタ7はミクサ(混合器)と呼ばれる，一種の入力受付けスイッチです．文の書きかたは

SOUND 7，〔排除するレジスタに割当てられた数値の合計〕

となっています．さて，各レジスタに割当てられた数値ですが

レジスタ0・1………1　　レジスタ2・3………2

レジスタ4・5………4

レジスタ6(レジスタ8に送るとき)………8

レジスタ6(レジスタ9に送るとき)………16

レジスタ6(レジスタ10に送るとき)………32

というわけで，数値の合計数は63になります．たとえばレジスタ4・5からの音を送りたかったら，排除したいのは(1＋2＋8＋16＋32)ですから，SOUND 7,59ということになります．違う表現をすれば(63－送りたいレジスタに割当てられた数値の合計)になります．

波形加工が効果音のきめ手

*ユーティリティプログラムをヒントにする

SOUND文の使いこなしのポイントは，波形加工用レジスタにあるといってもよいでしょう．さきほど説明した波形加工用発振器を使い，加工用レジスタ13を使って波形を指定します．波形は全部で8つあり，それぞれ数値で指定します（0〜15）．なおレジスタ8・9・10はすべて「16」を指定しないと，加工された音は出ません．

サンプルプログラムは，方形波およびノイズを音源として，波形を指定すると共に加工用の発振周波数を変えて，どのような音が出るかを試すものです．それぞれの数値をメモしておいて，本格的な効果音作りに役立てるとよいでしょう．このユーティリティプログラムを使って得たヒントの一例を書き出してみましょう．

波形	周波数	ノイズ周波数	加工周波数	出てくる音
10	高音		5Hz	虫の鳴き声
8	高音		5Hz	コップをたたく音
8	中音		5Hz	鐘の音
3		低音	5Hz	ピストルの発射音
8		低音	2Hz	工事現場の杭打ち
12		低音	2Hz	SLが速く走る音
10		低音	2Hz	SLがゆっくり走る音
10		高音	1Hz	波の音

※波形加工用レジスタと発振周波数

0.2Hz	CT=93	FT=0
0.5Hz	CT=37	FT=0
1Hz	CT=19	FT=0
2Hz	CT=9	FT=96
4Hz	CT=4	FT=173
6Hz	CT=3	FT=32
10Hz	CT=1	FT=224

※周期(period)：1回のくりかえしに必要な時間．周波数の逆数で，たとえば0.5Hzの周期は2秒，10Hzでは0.1秒になる

**音の止めかた

レジスタに書きこまれたデータは，別プログラムを実行（RUN）したり，リセットをかけて消さないかぎり，保存されているので，持続する音（レジスタ0〜6，レジスタ13を使ったもので持続波形を選んだとき）は，プログラム実行中，ずっと鳴っています．したがってこれらの音を消すためにはレジスタ8・9・10のデータを0にするか，発振器のレジスタのデータを0にしなければいけません．

時間を調整し，一定時間の音を出したあと，音を消す方法としてはFOR〜NEXTを使うほかに，ON TIME や ON INTERVAL文を使います．ゲームプログラムなどで，効果音を使うときは音の止めかたについてもよく考えておかないと，音が出ている間に他のことができなかったりしますので注意しましょう．

プログラム実行中の画面▶

```
100 '==================== ﾊｹｲｦ ｼﾃｲｼ ｼﾞｯｻｲﾉ ｵﾄ ｦ ﾀﾞｽ ======================
110 DEFINT A-Z:WIDTH 80,25:CLS
120 PRINT STRING$(78,"-"):'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ｹｲｾﾝｦ ﾋｸ
130 COLOR 5:PRINT"  ◣      ◢      ◣◣◣◣      ◣◢◣◢      ◣■■■      ◢◢◢◢      ◢■■■      ◢◣◢◣◢◣"
140 PRINT:  PRINT"  3      4      ■      10      11      12      13      14"
150 COLOR 7:PRINT STRING$(78,"-")
160 COLOR 5:PRINT:PRINT"CT=1・・・18Hz    CT=2・・・9Hz      CT=4・・・5Hz         CT=9・
・・2Hz"
170 PRINT:PRINT"CT=20・・・1Hz    CT=40・・・0.5Hz    CT=80・・・0.25Hz     CT=180・・・0.1H
z"
180 COLOR 7:PRINT STRING$(78,"-")
190 'ﾆｭｳﾘｮｸ
200 LOCATE 0,12
210 INPUT"ﾓﾄﾆｽﾙ ｵﾄ(1・・・ﾌﾂｳﾉｵﾄ  2・・・ﾉｲｽﾞ) ";MENU:IF MENU<1 OR MENU>2 THEN 210
220 ON MENU GOSUB 330,390:PRINT
230 INPUT"ﾊｹｲﾊ ﾄﾞﾚﾆ ｼﾏｽｶ(0・・・・13) ";WAVE:IF WAVE<0 OR WAVE>15 THEN 230
240 PRINT:INPUT"ｼｭｳｷﾉ ｾｯﾃｲ(CT:1・・・・255) ";CT:IF CT<0 OR CT>255 THEN 240
250 'ﾃﾞﾓ
260 SOUND 8,16:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾚｼﾞｽﾀ8 ｶﾗ ｵﾄｦﾀﾞｽ
270 SOUND 12,CT:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾊｯｼﾝﾖｳ ﾚｼﾞｽﾀ
280 SOUND 13,WAVE:'・・・・・・・・・・・・・・・・・・・・・・・・・・・・ﾊｹｲﾉ ｾﾝﾀｸ
290 PRINT:PRINT"ｺﾚｶﾞ ｼﾃｲﾉ ｵﾄﾃﾞｽ  ﾂｷﾞｦ ﾀﾒｽﾅﾗ ANYKEY PUSH"
300 IF INKEY$="" THEN 300 ELSE SOUND 8,0:CLS:RUN
310 '***** SQUARE WAVE SETTING
320 PRINT:INPUT"ｼｭｳﾊｽｳ(1・・・ﾃｲｵﾝ   2・・・ﾁｭｳｵﾝ  3・・・ｺｳｵﾝ) ";LMH
330 IF LMH<1 OR LMH>3 THEN 330
340 IF LMH=1 THEN FT=255 ELSE IF LMH=2 THEN FT=75 ELSE FT=20
350 SOUND 0,FT:SOUND 1,0:SOUND 7,(63-1):'・・・・・・・・・ﾊｯｼﾝｼｭｳﾊｽｳﾄ ﾚｼﾞｽﾀ7 ﾉ ｾｯﾃｲ
360 RETURN
370 '***** NOISE SETTING
380 PRINT:INPUT"ｼｭｳﾊｽｳ(1・・・ﾋｸｲ   2・・・ﾀｶｲ)";F:IF F<1 OR F>2 THEN 390
390 IF F=1 THEN NT=30 ELSE NT=1
400 SOUND 6,NT:SOUND 7,(63-8):'・・・・・・・・・・・・・・・・・ﾉｲｽﾞｼｭｳﾊｽｳﾄ ﾚｼﾞｽﾀ7 ﾉ ｾｯﾃｲ
410 RETURN
```

音の見本帖を作る

*一覧表の形式でデータを与える

いままでの話をまとめた，効果音の見本帖を作りましょう．SOUND文はレジスタを指定し，数値データを与えるという，ごく原始的な書き方の文になるので，プログラムを見ただけでは何がどうなっているのか分りにくいものです．そこで少しでも分りやすくするための一例として右のようなプログラムを作ってみました．

このプログラムではレジスタ0～レジスタ13に与えるデータをすべて書き出すようにし，しかもDATA文で与えるようにしています．プログラムによってはムダな部分が増えますが（データの0の部分が多くなってしまう），一覧表の形になっているので見やすくなります．

**サンプルとその解説

それぞれのレジスタに与えるデータは配列D(I)に読みこませ，演奏も同じように，FOR～NEXTの形式を使っているので，SOUND文そのものをいくつも書く必要はありません．行番号410のサブルーチンでキー入力待ちをし，入力がない間はひとつの効果音をずっと出しています．キー入力があると，レジスタ8・9・10の音量調整用のレジスタにすべて0が書きこまれ，音を止めてから次の効果音デモに移るようになっています．では，個々のルーチンを説明しましょう．

●ヘリコプター

ここではノイズを使い，波形加工用レジスタは12番を使った，くりかえし波形にしています．方形波から低い音を発振させると，また違った感じの音になります．

●踏切りの鐘

“カン・カン”と鳴る鐘の音です．3音を同時に発振させ，複雑な音色にしています．発振周波数をさらに細かく設定し，音を微妙にずらす（Aという音の2倍の周波数＋αをA′という音にする）と，よりリアルになるので，試してください．

●虫の鳴

“リーン・リーン”という感じの音です．音を出したり止めたり，周波数を変えたりすれば，さらに効果的です．

●オートバイ

エンジン音に似せて，低い音をずっと出したままにしています．

```
100 '============================= SOUND DEMO ==================================
110 '          8ｼｭﾙｲﾉ ｺｳｶｵﾝｦ ﾀﾞｼﾏｽ｡ﾂｷﾞﾉ ｵﾄｦ ｷｸﾅﾗ ANY KEY PUSH
120 DIM D(13):'0････13ﾉ ﾚｼﾞｽﾀ ﾃﾞｰﾀﾖｳ ﾊｲﾚﾂ
130 RESTORE 320:GOSUB 230:'ﾍﾘｺﾌﾟﾀ
140 RESTORE 330:GOSUB 230:'ﾌﾐｷﾘ
150 RESTORE 340:GOSUB 230:'ﾋﾟｽﾄﾙ
160 RESTORE 350:GOSUB 230:'ﾑｼ
170 RESTORE 360:GOSUB 230:'ｵｰﾄﾊﾞｲ
180 RESTORE 370:GOSUB 230:'ﾅﾐ1
190 RESTORE 380:GOSUB 230:'ﾅﾐ2
200 RESTORE 390:GOSUB 230:'ｺｳｼﾞ ｹﾞﾝﾊﾞﾉ ｸｲｳﾁ
210 FOR I=0 TO 13:SOUND I,0:NEXT:END
220 '
230 FOR I=0 TO 13:READ D(I):NEXT:GOSUB 240:GOSUB 410:'DATA ｦ ﾖﾐ ｼﾞｬｺｳ
240 FOR I=0 TO 13:SOUND I,D(I):NEXT:RETURN:'ｴﾝｿｳ ｼﾞｯｼ SUB
250 '
260 '--------- ﾚ  ｼﾞ  ｽ  ﾀ  ﾊﾞ  ﾝ  ｺﾞ  ｳ  ﾄ    ｱ  ﾀ  ｴ  ﾙ  ﾃﾞ  ｰ  ﾀ  ---------
270 '     0  1   2   3   4   5      6       7       8   9  10     11   12       13
280 '--------------------------------------------------------------------------
290 '     FT CT  FT  CT  FT  CT     NT      7       8   9  10     FT   CT       13
300 '     └---- FREQ--------┘   NOISE  MIXER      VOLUME          FREQ          ﾊｹｲ
310 '--------------------------------------------------------------------------
320 DATA 0,0,0,0,0,0,            31,     55,      16,0,0,       200,0,      12
330 DATA 100,0,50,0,20,0,         0,     56,      16,16,16,       0,10,      8
340 DATA 0,0,0,0,0,0,            15,     55,      16,0,0,       120,10,      3
350 DATA 60,0,30,0,0,0,           0,     60,      16,16,0,      173,0,      10
360 DATA 0,14,0,0,0,0,           28,     56,      10,0,0,         0,0,       0
370 DATA 0,0,0,0,0,0,             5,     55,      16,0,0,         0,150,     8
380 DATA 0,0,0,0,0,0,            15,     55,      16,0,0,         0,150,     8
390 DATA 0,0,0,0,0,0,            31,     55,      16,0,0,         0,20,      8
400 '
410 WHILE INKEY$="":WEND:'･･････････････････････････ｲﾁｼﾞ ﾃｲｼ
420 FOR REG=8 TO 10:SOUND REG,0:NEXT:'･･････････････ｵﾄｦ ｹｽ
430 RETURN
```

***より効果のある演出を考えよう

いまのプログラムをもとにして，より効果的な音の演出をするためにはどうしたらよいでしょうか？

- **時間の調整**：音を出す・止めることのくりかえしを，自然な感じにするため，演奏時間をじょうずに設定します．
- **音の演出**：単一の効果音にしないで，出てくる音にひとつの物語を与える（SLなら発車一徐行一急行一坂をのぼる一停止させるなど）と，より親しみのあるものになります．
- **偶然性を持たせる**：発振周波数を乱数によってコントロールするなどの工夫で，自然界の音に近くなります．

PLAY文の基礎

＊PLAY文の基本は文字列による命令

音楽演奏用のPLAY文はマクロ言語（macro language）になっており，命令は一連の文字列によって与えられます．まず，それをまとめましょう．

- **●音階**：C，D，E，F，G，A，Bとオクターブ（O1～O8）
- **●半音階**：＋または－を音符のあとにつける
- **●音の長さ**：L1～L64，付点は音符のあとにピリオドをつける
- **●休止符**：R1～R64，付点は数値のあとにRを重ね書きする
- **●テンポ**：T32～T255
- **●音量**：V0～V15
- **●波形**：S0～S15，周期はM0～M65535
- **●別の音階表示法**：N1～N96で音階を表示

FM-77は，PLAY文のあとに書かれた文字列を読んで，その指示に従いますが，デフォルト値はつぎのように設定されているので，「書かれていない」部分はデフォルト値に従って解釈します．

- **●オクターブ指定**：O4（ラ＝Aの音を440Hzとする）
- **●テンポ**：T100
- **●音の長さ**：4分音符に相当（L4）
- **●音量**：中間（V8）

※付点音符や付点休符は，表わされた音符の1.5倍の長さを意味する．たとえば4分音符を4の長さとし，2分音符を8の長さとすると，付点4分音符は6の長さになる．

したがって

```
PLAY "ec" は，PLAY "V8T100L4O4EC"
```

とおなじです．

＊＊PLAY文におけるデフォルト値

ところでPLAY文で書いたいろいろな命令は，FM-77をリセットしないかぎり内部に残っています．通常の変数は"NEW"あるいは"RUN"ですべて0（数値変数）かヌル（文字列変数）になるのですが，SOUND文，PLAY文でセットした命令はこの法則があてはまりません．したがって

```
100 PLAY "V3C"
```

としたあと， NEW としていったんプログラムを消して

```
100 PLAY "C"
```

という文を書いても，新しいプログラムの音量については，もともと

のデフォルト値である「V8」ではなく，ひとつ前のプログラムの文で指定した「V3」がそのまま適用されてしまいます．

もうひとつ，PLAY文では変数を使う書式があります．

マクロ用命令＝変数；

という書法によるもので，たとえばVOLという変数に6を代入し

PLAY "V＝VOL;"

というPLAY文を書けば，これは

PLAY "V6"

とおなじことになります．

***PLAY文のいくつかの書きかた

PLAY文の基本は文字列操作なので，この操作さえまちがえなければ，いろいろな書きかたが許されます．サンプルでその代表例を見ましょう．

行番号120はもっともオーソドックスなもので，いわば文字列定数の形で演奏を指示しています．なおサブルーチンは初期設定値に戻すためのもので，本質的には不要ですが，一種の安全装置になります．

行番号150はREAD～DATAによるもので，変数に音階を与えています．行番号190もおなじような使いかたです．

行番号230は，音量とテンポをひとつの変数（VOL）でコントロールしている例です．

```
100 '================== PLAYﾌﾞﾝﾉ ｷﾎﾝ / PLAYﾌﾞﾝﾉ ｶｷｶﾀ ================
110 '* ﾁｮｸｾﾂ ｼﾃｲ
120    GOSUB 270:PLAY"O4 cdef":PLAY"R1"
130 '
140 '** READ DATAｶﾞﾀ
150    GOSUB 270:READ MUSIC$:PLAY MUSIC$+"R1"
160  DATA cdef
170 '
180 '*** ﾓｼﾞﾚﾂ ｼﾃｲ
190    GOSUB 270:DOREMI$="cde":PLAY DOREMI$+"R1"
200 '
210 '**** ﾍﾝｽｳ ﾀﾞｲﾆｭｳ
220   GOSUB 270
230 FOR VOL=6 TO 10:TEMPO=VOL*25:PLAY"V=VOL;"+"T=TEMPO;"+DOREMI$:NEXT
240 END
250 '
260 'PLAYﾌﾞﾝﾗ ﾋｮｳｼﾞｭﾝﾃｷﾅ ﾓﾉﾆ ﾓﾄﾞｽ SUB
270     PLAY"O4L4R4T100V8":RETURN
```

“S”コマンドがPLAY文のキーポイント

＊音と音のつながりをふせぐには

楽譜があれば，それを忠実にPLAY文に移すことで基本的な音楽演奏は可能になります．ただしひとつだけ注意しなければいけないこととして，たとえば“ドドド”と3つの音が続くとき，波形指定なしの演奏ではこの音はつながってしまい，ひとつの“ド”にしか聞えません．楽器の場合は音がだんだん弱くなったりすることが多いし，たとえばオルガンでも“ド”を弾いて指を離せばそこで音は途切れるためにちゃんと3つの音になります．

これをさけるためにはPLAY文を

“ド・短かい休符・ド・短かい休符・ド”

というように書くか，波形指定をして，時間と共に音量が変るようにします．波形とその周期（周波数）の指定はSOUND文とよく似ています．波形についてはまったくおなじです．周期については

PLAY “M○○○”

とし，○○○に数値を入れます．数値は16ビット分(0～65535)で，発振周波数はN＝1～65535として

$f=122880\div(256\times N)=480\div N$ (Hz)

になります．波形指定をすれば，音階がおなじでも音がつながって聞えることはありません．

＊＊ユーティリティプログラムで，目的の音色を探す

サンプルプログラムは“ドレミファソラシド”を3つのテンポで演奏させ，波形と周期の設定によってどう変わるかを試すものです．何回か実行させ，目的の音色に近いものがあったらメモをしておきます．なお，テンポによって聞えかたがずいぶん違いますし，場合によっては音がひどく小さい（例：テンポが速いのに，周期が遅い……発振周波数が低いと，十分な音量で演奏する前に，つぎの音符を演奏する）こともあるので，必らず3つのテンポで確認してください．

なお，このプログラムでは波形指定なしを，SHAPEに“16”を代入した時にしています．この時は周期入力を飛ばしてすぐに演奏させています．また，演奏実行の間，FM-77は他に何もしないで演奏の終了を待っているわけではありません．ここでは演奏音と画面に出力する文字列がおなじタイミングになるよう，工夫しています．

＊＊＊“S8M3000”などが万能型命令

いまのユーティリティプログラムで，いろいろな音が出ることが分りますが，曲を仕上げる前に音符通りに演奏しているかどうかなどを知る時は，たとえば“S8M3000”とすると，ピアノのような音で演奏してくれるし，すべての音がつながらずに聞えるので，これを標準的な波形にします．なお，それ以外では

“S3M1000”……スタッカート（音をひとつひとつ区切る）

“S14M500”……ビブラート（遅い曲で有効）

“S12M400”……トレモロ

などが利用価値のある波形です．

```
100 '======================== ｴﾝｿｳﾊｹｲ ﾄ ｼﾞｯｻｲﾉ ｵﾝｶﾞｸ ========================
110 WIDTH 80,25:CLS
120 PRINT STRING$(78,"-"):'·································ｹｲｾﾝｦ ﾋｸ
130 COLOR 5:PRINT"  ◣     ◢     ◣◣◣◣    ◣◢◣◢    ◣■■■    ◢◢◢◢    ◢■■■    ◢◣◢◣◢◣
";:COLOR 6:PRINT" ■■■■■ ":COLOR 5
140 PRINT:  PRINT"  3      4      8       10      11      12      13       14
";:COLOR 6:PRINT"  16"
150 COLOR 7:PRINT STRING$(78,"-")
160 COLOR 5:PRINT:PRINT"M=48···100Hz    M=480···10Hz    M=4800···1Hz   M=9600···
.5Hz   M=48000···0.1Hz "
170 COLOR 7:PRINT STRING$(78,"-")
180 'ﾆｭｳﾘｮｸ
190 PRINT:PRINT
200 INPUT"ﾊｹｲ ﾊ ﾄﾞﾚｦ ｴﾗﾋﾞﾏｽｶ(0···15)  ";SHAPE
210 IF SHAPE=16 THEN PLAY"V10":GOTO 250:'··············ﾊｹｲﾉ ｼﾃｲｶﾞ ﾅｲﾄｷ
220 INPUT"ｼｭｳｷ ﾊ ｲｸﾂﾆ ｼﾏｽｶ(1···65535) ";CYCLE
230 PLAY"S=SHAPE;M=CYCLE;"
240 'ｴﾝｿｳ1··············ﾕｯｸﾘﾄｼﾀ ﾃﾝﾎﾟ
250 PRINT:TPLAY=60
260 PRINT"TEMPO=";TPLAY;" L4O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4"
270 PLAY"T=TPLAY; L4O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4":GOSUB 400
280 'ｴﾝｿｳ2·············ﾋｮｳｼﾞｭﾝ ﾃﾝﾎﾟ
290 PRINT:TPLAY=120
300 PRINT"TEMPO=";TPLAY;" L4O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4"
310 PLAY"T=TPLAY; L4O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4":GOSUB 400
320 'ｴﾝｿｳ3·············ﾊﾔｲ ﾃﾝﾎﾟ
330 PRINT:TPLAY=220
340 PRINT"TEMPO=";TPLAY;" L16O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4"
350 PLAY"T=TPLAY; L16O4 ccddeeffggaabb  O5 c R4 O5 c  O4 bagfedc R4":GOSUB 400
360 '
370 FOR W=1 TO 2000:NEXT:GOTO 110
380 '
390 'ｴﾝｿｳｼﾞｶﾝﾄ ｶﾞﾒﾝｼｭﾂﾘｮｸｦ ｺﾝﾄﾛｰﾙｽﾙ SUB
400 T=INT(1.3E+06)/TPLAY:FOR W=1 TO T:NEXT:RETURN
```

和音と伴奏のつけかた

*単純な曲も伴奏で見違えるように

FM-77は同時に3つの音を独立して出すことができますが，それぞれのパート（声部）※の音量を変えたいとき（主旋律は大きめの音で，伴奏は小さめの音で演奏するなど）は，波形指定はできません．なお波形指定をして3つの音を同時に出すと，音はにごり気味で，きたなくなりますから，曲によって波形加工をするかどうかを使い分けます．

簡単な曲であっても，それに伴奏をつけると見違えるほど立派になりますから，たとえばエレクトーンやギター用の楽譜などを参考にして，2音や3音を使うとよいでしょう．

**楽譜がないときの伴奏のつけかた

自分で作った曲，あるいは楽譜があっても伴奏のパートが書かれていないものなどは自分で工夫をしなければいけません．いちばん簡単なのは，協和音をつけてゆく方法です．

- 完全8度：ちょうど1オクターブ離れている音．主旋律が“ドレミ”なら，それより1オクターブ低い“ドレミ”を演奏します．
- 長3度：主旋律が“ミ”ならそこから2音低い“ド”を演奏すると長3度の関係になります．この協和音程が万能といえます．
- 完全5度：主旋律が“ソ”なら，4音低い“ド”といった関係．
- 完全1度：いわゆる“ユニゾン”と呼ばれるものです．本物のユニゾンは，音量が増し，力強い音楽の表現になりますが，FM-77ではその効果はほとんど出ません．

もうひとつの方法は，ギターの伴奏によく使われる，コードでリズムを刻む方法です．ただし一般のコードは最大で3音を同時に鳴らしたりしますが，FM-77では3音を伴奏には使えない（主旋律の分がなくなってしまう）ので，2音でコードを演奏してゆきます．コードは何種類もあり，くわしいことは専門書を見ていただくとして，ここではいくつかの例を紹介しましょう．

- C：オクターブ低い“ド”，オクターブ上の“ド”と“ミ”
  3拍子の曲なら，低い“ド”をひとつ，高い“ド”・“ミ”を2つ演奏する（つまり「ドン　パッ　パッ」になる）といった使い方
- G：“ソ”，“シ”，“レ”の組合せ
- F：“ファ”，“ド”，“ラ”の組合せ

※PLAY文などには大文字も小文字も使える．音階は小文字に，PLAY命令は大文字にすると，文字列が続いていても，プログラムリストが読みやすくなる

```
100 '================ ｼﾞﾝｸﾞﾙ ﾍﾞﾙ  =======================
110 PLAY"T100","T100"
120 READ M1$,M2$
130       IF M1$="END" THEN END
140       PLAY M1$,M2$
150 GOTO 120
160 '
170 '"** ﾒﾛﾃﾞｨ **  "            "  ** ﾊﾞﾝｿｳ **  "
180 DATA "V15","V12"
190 DATA "L8O4cagfc4.R8",      "L8O2fO3cO2fO3cO2fO3cO2fO3c":      'ﾉｦｺｴﾃ
200 DATA "L8O4cagfd4.R8",      "L8O2fO3cO2fO3cO2b-O3dO2b-O3d":    'ｵｶｦｺｴ
210 DATA "L8O4db-age4.R8",     "L8O2b-O3dO2b-O3dcece":            'ﾕｷｦ ｱﾋﾞ
220 DATA "L8O5dcO4b-ga4.R8",   "L8O3cecefcO3aO3c":                'ｿﾘﾊ ﾊｼﾙ
230 DATA "L8O4cagfc4.R8",      "L8O2fO3cO2fO3cO2fO3cO2fO3c":      'ﾀｶﾗｶﾆ
240 DATA "L8O4cagfd4.R8",      "L8O2fO3cO2fO3cO2b-O3dO2b-O3d":    'ｺｴｱﾜｾ
250 '
260 DATA "S13M400",            "S13M400"
270 DATA "L8O4db-agO5cccc",    "L8O2 b-O3dO2b-O3dcece":           'ｳﾀｴﾔ ﾀﾉｼｲ
280 DATA "L8O5dcO4b-gf4.R8",   "L8O2 b-O3cegf4.R8":               'ｿﾘﾉｳﾀ
290 '
300 DATA "T105S3M1500",        "T105S3M1500"
310 DATA "O5L8aaa4aaa4",       "O4L8fff4fff4":          'ｼﾞﾝｸﾞﾙ ﾍﾞﾙ   ｼﾞﾝｸﾞﾙﾍﾞﾙ
320 DATA "O5L8aO6cO5f.g16a4R8","O4L8fac.e16fR8":        'ｽｽﾞｶﾞ ﾅﾙ
330 DATA "O5L8b-b-b-4b-aaa",   "O4L8ddd4dfff":          'ｿﾘｦ ﾄﾊﾞｼﾃ
340 DATA "O5L8aggfgO6cc4",     "O4L8feedeee4":          'ｳﾀｴﾔ ｳﾀｴ
350 DATA "O5L8aaa4aaa4",       "O4L8fff4fff4":          'ｼﾞﾝｸﾞﾙ ﾍﾞﾙ   ｼﾞﾝｸﾞﾙﾍﾞﾙ
360 DATA "O5L8aO6cO5f.g16a4R8","O4L8fac.e16fR8":        'ｽｽﾞｶﾞﾅﾙ
370 DATA "O5L8b-b-b-4b-aaa",   "O4L8ddd4dfff":          'ｳﾏｦ ﾄﾊﾞｾﾃ
380 DATA "O6L8ccO5b-gf4.",     "O4L8eegef4.":           'ｲｻﾞｳﾀｴ
390 DATA END,END
```

＊＊＊ジングルベルを例にして

上のプログラムはジングルベルを２声を使って演奏させる例です．全音楽譜出版のスコアを基本にしています（竹田由彦編）．演奏の表情づけは３つに分れており，行番号180でオーソドックスなものを，行番号260で３度，５度の協和音を，行番号300からは鈴の音の感じになる波形を選び，音階も１オクターブ上げています．なお最初の部分の伴奏は，たとえば“ファ・ド・ファ・ド……”といったように，８分音符でこまかく進行していますが，主旋律の音符（休止符も含む）の総合計と，伴奏の音符の総合計を合わせないといけません．自分で勝手な長さの音符を使うと，あとで拍数などが合わなくなりますからよく注意しましょう．なお，最後の“ジングルベル……”からのコーダに近い部分は，わずかにテンポを速くしました．

FM-77を鍵盤楽器にする

＊黒鍵もついた本格派

FM-77のキーボードを楽器の鍵盤代りにして，作曲の手伝いや演奏そのものが楽しめるようにしましょう．このプログラムで作った鍵盤は音域が十分とはいえませんが，簡単な曲ならたいていのものが演奏できます．まずリストの450行以降を見てください．

中央の“C”はカタカナの“ソ”に設定しています．そしてこのキー配列の上段を半音のキー（黒鍵）として使っています．なお高音部はテンキーも使っており，行番号500の“1，2，3”はテンキー部です．

＊＊INSTR関数を使う

あるキーを押したとき，あらかじめ設定した音を出す方法はいくつかが考えられます．ここではキー配列にしたがった文字列を用意し，インストリング関数とMID＄関数の組合せで音を出します．たとえばカタカナの“ソ”を押したときを考えましょう．M1＄の3番目に“ソ”があるので，行番号220のPは「3」になります．行番号260でPLY＄はMU1＄（“abcde…”）の中から1文字を探します．MID＄（MU1＄，P, 1)…つまりMU1＄の左からP番目の文字を1個取出して，これをPLY＄とするので，いまの場合は3番目，結果的にPL＄は“C”になるというわけです．

行番号230～250で，Pの値によってオクターブ値を決めているので，実際のPLY＄は“O4”＋“c”になるのですが，カタカナの文字から音階を決めているのが行番号220～260です．

なお半音についても同じ考えで処理していますが，ここではPの値が0のとき（M1＄の中に，押したキーの文字がないとき）に，半音のキーを押したかどうかを調べて(行番号320)，押していれば半音を上げる処理をします．

なお，押したキーがどの音階なのかを画面に表示して，あとでプログラムの文中に生かせるようにしていることと，小節の区切りなどの目印になるように，“＊”マークを自由な位置に入れるようにもしています．配列を使って，押したキーの音階をメモリに入れておけば再演奏もできます．音階の訂正ルーチンなども作ればより実用的になるので工夫をしてみましょう．“＊”はテンキー部にあるものを使うとよいでしょう．

なお，音域を広げるための工夫としては，FM-77のキーボードを何段も使ったり，カナ／アルファベットの切換えを使う（“ソ”で中央のc，“C”にしたらオクターブ低いcにする）方法があります．

```
100 '==================== FM-77ﾉ KEY BOAD ｦ ｹﾝﾊﾞﾝﾆ ｽﾙ ========================
110 '=                    KEY BOAD ｦ ｶﾀｶﾅ ﾓｰﾄﾞﾆ ｼﾃｸﾀﾞｻｲ                           =
120 '=      ﾃﾞｰﾀｶﾞ ｶﾞﾒﾝ ｲｯﾊﾟｲﾆ ﾅﾙﾄ ｼｮｳｷｮｻﾚﾏｽ    * ﾏｰｸｦ ｵｽﾄ ｸｷﾞﾘﾉ ﾒｼﾞﾙｼﾆ ﾅﾘﾏｽ    =
130 '=====================================================================
140 '
150 C=0:WIDTH 40,25:CLS:PLAY"V10L8"
160 GOSUB 440:'................................ｹﾝﾊﾞﾝﾉ ﾃﾞｨｽﾌﾟﾚｲ
170 '------------------------- ﾓｼﾞﾚﾂﾄ ｵﾝｶｲﾉ ﾃｲｷﾞ---------------------------------
180 M1$="ﾂｻｿﾋｺﾐﾓﾈﾙﾒﾛ123":MU1$="abcdefgabcdefg"
190 M2$="ﾄﾊｷﾏﾉﾘｹﾑ":MU2$="acdfgacd":'........................ﾊﾝｵﾝｶｲﾖｳ
200 '------------------------- KEYﾎﾞｰﾄﾞｶﾗﾉ ﾆｭｳﾘｮｸ-------------------------------
210 A$=INKEY$:IF A$="" THEN 210 ELSE IF A$="*" THEN PLY$="*":GOTO 280
220 P=INSTR(M1$,A$):IF P=0 THEN GOSUB 320:'................ﾊﾝｵﾝｶｲ ｶ ﾄﾞｳｶﾉ ﾊﾝﾀﾞﾝ
230   IF P>=10 THEN O$="O5"
240   IF P<=2 THEN  O$="O3"
250   IF P>2 AND P<10 THEN O$="O4"
260 PLY$=O$+MID$(MU1$,P,1):PLAY PLY$
270 '--------------------------- ﾃﾞｰﾀ ｼｭﾂﾘｮｸ ---------------------------------
280   PRINT  USING"&   &";PLY$;
290   C=C+1:IF C>=120 THEN GOSUB 400:C=0:'.........ｶﾞﾒﾝ ｼｭﾂﾘｮｸﾉ ﾃﾞｰﾀﾉ ﾘｮｳｦ ﾊﾝﾀﾞﾝ
300 GOTO 210
310 '============================= ﾊﾝｵﾝｶｲ=================================
320 P=INSTR(M2$,A$):IF P=0 THEN RETURN 210
330   IF P>=7 THEN O$="O5"
340   IF P<=1 THEN O$="O3"
350   IF P>1 AND P<7 THEN O$="O4"
360 PLY$=O$+MID$(MU2$,P,1)+"#":PLAY PLY$
370   RETURN 280
380 '
390 'ｶﾞﾒﾝ ｼｮｳｷｮ
400 FOR Y=0 TO 15:LOCATE 0,Y:PRINT STRING$(80," ");:NEXT:LOCATE 0,0
410 RETURN
420 '
430 '***** ｹﾝﾊﾞﾝﾉ ｻｸｾｲ
440 COLOR 6:LOCATE 0,17
450 PRINT TAB(5);"  ﾄ   ﾊ ｷ   ﾏ ﾉ ﾘ   ｹ ﾑ"
460 PRINT TAB(5);"| ■ | ■ ■ | ■ ■ ■ | ■ ■ | | |"
470 PRINT TAB(5);"| ■ | ■ ■ | ■ ■ ■ | ■ ■ | | |"
480 PRINT TAB(5);"|a|b|";:COLOR 5:PRINT"c|d|e|f|g|a|b";:COLOR 6:PRINT"|c|d|e|f|g|"
490 PRINT TAB(5);"| | | | | | | | | | | | | | |"
500 PRINT TAB(5);"|ﾂ|ｻ|ｿ|ﾋ|ｺ|ﾐ|ﾓ|ﾈ|ﾙ|ﾒ|ﾛ|1|2|3|";
510 PRINT TAB(5);"└┴┴┴┴┴┴┴┴┴┴┴┴┴┘";
520 COLOR 7:LOCATE 0,0
530 RETURN
```

乱数を使って即興演奏をさせる

＊乱数を使ったフリーミュージック

本物の即興演奏は，演奏者の感性によりますが，コンピュータではプログラムによって，たとえば明るい感じの曲なら長調を使い，テンポも早めにするなどを考えれば，多少は曲想が設定できるものの，芸術的なものを作るのは難かしいものです．

ここに紹介するものは，まったくのフリー演奏で，小節数，音階，音長などをすべて乱数によって作り出しています．小節数，音長などは1，2，4，8，16…が基本ですから，ここでは1，2，3…といった数値を使ってべき乗計算によって求めます(行番号140，160)．

なお音階は“c，d，e…”といったアルファベット系列以外に数値による指定ができるので，ここでは，N＝30～69までを発生させます(行番号160)．N＝48が中央のド（C）ですから，約3オクターブの音域です．このプログラムではまず小節数と演奏するときの音量を決めてから，音階などを配列にしまいます．

演奏をさせるルーチンは行番号190～210ですが，演奏と共に四角が画面に出てくるサブルーチンを呼び，にぎやかにしています．また設定した小節の演奏が終ると，休止するようにしました(行番号220)．

なお，音階の指定その他を工夫すると人間が普通に演奏していて，しかも即興演奏のように聞える曲になるので工夫してみましょう．

```
100 '====================== RANDOM ACCESS MUSIC ===============================
110 DIM DOREMI(16),NAGASA(16),C(16)
120 IF TIME>30000 THEN TIME$="00:00:00"
130 CLS:RANDOMIZE TIME
140 XS=INT(RND*5):XS=2^XS:VOL=INT(RND*5)+10:'XS=ｼｮｳｾﾂ(1,2,4,8,16)
150 FOR I=1 TO XS
160   C(I)=(RND*7)+1:DOREMI(I)=INT(RND*40)+20:X=INT(RND*5):NAGASA(I)=2^X
170 NEXT
180 PLAY"V=VOL;"
190 FOR I=1 TO XS
200    PLAY"N=DOREMI(I);L=NAGASA(I);"   :GOSUB 250
210 NEXT
220 REST=NAGASA(1):PLAY "R=REST;"
230 GOTO 120
240 '
250 X=NAGASA(I)+I*30:Y=150-DOREMI(I)*2:LINE(X,Y)-(X+30,Y+20),PSET,C(I),BF
260 FOR T=0 TO 500:NEXT:RETURN
```

8

これからの言語

Logoを研究しよう

Logoが問題むきの言語といわれるわけは…

*本質的にはLogoはやさしい言語

Logoはアメリカのマサチューセッツ工科大学のMIT（マサチューセッツ工科大学）で生まれた言語です．英語のロゴグラム（logogram：円を￥と表わすような符号化した言葉）などからも分るように，人間がふだん使っている言葉でコンピュータを使いこなすことを目的として開発されました．

ただ，生まれがアメリカですから，コンピュータへの命令は英語が使われる点で，日本人にはちょっと苦手な部分があります．たとえばBUTFIRST（バットファースト）というLogoの命令は，文字のつづりや単語の中の「最初以外の」ものを処理するときに使いますが，英語圏の人々にはこの命令が感覚的にすぐに分るのに対し，英語圏以外の人々にはその意味をまず頭の中に入れなければいけない，ということになります．Logoを使うとき，日本人が「どうもとりつきにくい」などと思うのは，Logoの，コンピュータとしての本質的な動作とこれらの英語の命令とを混同してしまうことが多いからにほかなりません．

※Logoを開発したのはMITのS．パパート，スイスの教育心理学者のJ．ピアジェなどである．子供にも扱えるプログラム言語，というのが開発のひとつの動機といわれている

**手続きと定義とは

人工知能（artificial intelligence）の分野にもLogoは活躍します．それはなぜでしょう．またLogoによるグラフィックスは小学生程度にも十分に使えるといわれます．それはなぜでしょうか．使い手がコンピュータにいろいろな命令を伝えるとき，Logoではその内容を使い手がコンピュータに教えることができるからです．

たとえば「朝」ということを，私がこのように考えているとしましょうか．

朝は「起きる」こと「歯をみがく」こと「パンを食べる」こと．

Logoでは，それぞれの内容を，手続きを経て定義できます．

```
TO　起きる　（←定義のタイトル）
目ざましが鳴る　フトンから出る　着がえる　（←手続きの内容）
END　　　　（←手続き終了）
```

これとおなじようにして「歯をみがく」「パンを食べる」を定義し

```
TO　朝
起きる　歯をみがく　パンを食べる
END
```

として，朝ということをLogoを通じてコンピュータに伝えるのです．

※手続き：プロシジャ

これでコンピュータには使い手が考える「朝」が教えこまれました．このことは，見方を変えればコンピュータには「朝」という命令がひとつ，新しく増えたのとおなじになります．これをディスクにしまっておけば，いつでも自由に呼び出せるので，たとえばBASICのように行番号の何番までがどのようなルーチンであったのか，といったことに頭を悩ませる必要もありませんし，Aというプログラムでも Bというプログラムでも，共通していつでも自由に呼び出せることになります．

※Logoはまたロゴス（logos：理性）にも通じている．考えを表わすひとつの方法が言葉であり，コンピュータを操作することは使い手の考えを，記号やプログラム言語を使って具現することにほかならない

＊＊＊再帰的なプログラム

さて，いまの「朝」の定義を，もしつぎの形で実行したらどうなるでしょうか

TO　朝
起きる　歯をみがく　パンを食べる　朝
END

コンピュータは起きる・歯をみがく・パンを食べるを実行したあとに朝にぶつかります．では「朝」はどう定義されていたでしょうか？

朝は「TO　朝～END」の中で「起きる　歯をみがく　パンを食べる　朝」と定義されていました．

このプログラムは，結果的には起きる・歯をみがく・パンを食べる
起きる・歯をみがく・パンを食べる…………のくりかえしになります．これが「再帰」あるいは「循環」とよばれるものです．

Logoでは再帰的なプログラムを書くことができるのですが，考えれば考えるほどふしぎだ・おかしい・理解できない，という人がいるに違いありません．それはあなたがまだLogoを使っていないからです．少しの時間であってもLogoに触れるとこのことは特に意識しなくなります．Logoは，COBOLやBASICなどのように，こまごまとした書きかたにあまり制約されず，○○ということは△△ということを意味する，と思ったら○○の部分を手続きを経て定義して，こうした部分をどんどん作ってゆく，といった操作が可能なのです．その代り，A，B，C，D…といった多くの数値を計算し，出力する数値の桁数をそろえてプリンタに送る，といったことは得意ではありません．Logoが《問題むき》…なにかの命題を解決するのにむいている…言語とよばれるのはこのためです．

前置きはさておき，さっそくLogoの世界をのぞいてみましょう．

※再帰：リカージョン

まず，四角は四角になるようにしよう

＊さいしょのプログラム

まず，ディスプレイのタテ・ヨコ比を正しい関係にしましょう．フォーマットずみの，書きこみ可能なディスケットをドライブに入れてください．そして？マークが出たらテキストスクリーンにし

?TO SHIKAKU	(RETURN)	定義をしたい
TO SHIKAKU	(RETURN)	定義はSHIKAKU
HT	(RETURN)	タートルを消す
FD 150	(RETURN)	前へ150
RT 90	(RETURN)	右へ90度
SHIKAKU	(RETURN)	SHIKAKUを呼ぶ
END	(RETURN)	定義終り
	(ESC)	手続きから？へと，もどる

とします．これで正方形を描くための準備ができました．SSにし，

※TS：テキストスクリーン

※SS：スプリットスクリーン

?SHIKAKU　　(RETURN)

で画面には四角が描かれるでしょう．ただしこのプログラムは前のページで説明した再帰的なプログラムですから，BREAKしないかぎり，永久に四角を描いています．さて，この正方形は，真四角になっていましたか？　SAVE “DO：SHIKAKUとします．

＊＊タテ・ヨコ比を補正してあげましょう

真四角になっていないときは，HOSEIを作ってください．キー操作はいま述べたのとおなじ方法です．私の場合は手続き中の数値が「.435」でよかったのですが，あなたのディスプレイに合せてこの数値を変えてやります．正方形が正方形に見えるまで，SHIKAKUを実行し，たしかめたあとに

?TO HOSEI　　(RETURN)

で補正の定義を画面に出し，数値を最適なものにしてやります．

これも　SAVE “DO：HOSEI　でディスケットにおさめましょう．SAVEが終ると画面にメッセージが出て，2つの手続きが定義ずみで，それがディスケットにおさめられていることが分ります．BASICの感覚からすれば，いまは「補正」のプログラムだけが“HOSEI”の名前でSAVEされたはずなのに，Logoでは「補正」と「四角」の2つがおさめられたのです．

```
TO HOSEI
.SETSCRUNCH 0.435
END
```

*****ワークスペースをすべて呑みこむSAVE**

その証拠に「補正」をSAVEしたあとに画面のメッセージは

2 PROCEDURES SAVED

となったはずです．あなたはいま，「補正」だけをHOSEIという名前でディスケットにおさめたと思っているのに，です．ためしに

?ERPS （すべてのプロシジャを消せ）

としてリターンキーを押したあとに

LOAD "DO：HOSEI （RETURN）

としてください．

HOSEI DEFINED （補正は定義ずみ）

SHIKAKU DEFINED （四角は定義ずみ）

というメッセージが出ます．

Logo では SAVE や LOAD ERPS などの，Logoに最初から備わっている命令をプリミティブ（primitive）といいます．初期命令，あるいは固定命令とでもいえるでしょう．これに対しいま定義した「四角」や「補正」はワークスペースと呼ばれるところにしまわれ，SAVE命令によってワークスペースにある手続きはすべてディスケットにしまわれます．BASICでいえば，ひとつの行番号の文が，ひとつの定義に相当する，といったように考えてもよいでしょう．したがって「補正」の手続きを定義することと，SAVEして“HOSEI”をディスケットにしまうこととは別なのです．

******部分だけをSAVEする方法は**

ひとつひとつの手続きは部分で，SAVEするときはワークスペースにあるもの全体がディスケットにおさめられることを知りましょう．なお，“HOSEI”には「補正」だけをSAVEしたかったら

ER [SHIKAKU] （SHIKAKUを消す）

ERF "DO：HOSEI

SAVE "DO：HOSEI

の3つの操作をじゅんばんにおこなってください．BASIC の感覚でいえば「四角」にあたる文の行番号を消して，すでに“HOSEI”の名前でSAVEしてあったプログラムをKILLしてから，もういちどSAVEしなおす作業とよく似ています．

Logoに慣れるためにも マスターディスケットによる起動→手続き→SAVE→マスターディスケットによる起動→LOADなどを何回かくりかえしてみましょう．

※ERF：イレースファイル

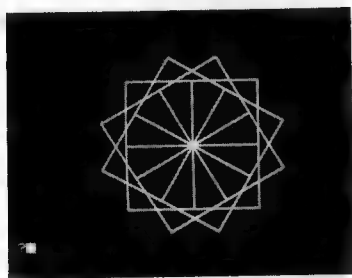
四角を作り，回転させる

＊準備の手続き，手続きの組合せ

```
TO KAITENSHIKAKU
JUNBI
REPEAT 12 [SHIKAKU120 RT
                      30]
END
TO SHIKAKU120
REPEAT 4 [FD 120 RT 90]
END

TO JUNBI
HOSEI
SS
CS
HT SETPC 7
END

TO HOSEI
.SETSCRUNCH 0.435
END
```

▲回転する四角の手続き

▼COLOR7，そしてゆっくりと描く

```
TO KAITENSHIKAKU
SETPC 7
JUNBI
REPEAT 12 [SHIKAKU120 WAIT 50 RT 30]
END
```

いちど電気を切り，最初からスタートしましょう．まず

LOAD "DO：HOSEI

を実行し，つぎに「準備」の手続きを定義します．この手続きはプリミティブ（CS：クリアスクリーン，HT：ハイドタートル，SS：スプリットスクリーン）や「補正」という，画面への出力準備のための手続きからできています．

つぎに一辺が120の長さの四角を描く手続きを定義します．それがSHIKAKU120です．この手続きはすぐに分るでしょう．

REPEAT 4 [FD 120 RT 90]

4回のくりかえし [前へ　120　右へ　90度] ということです．

さて，これで下ごしらえはできました．この4角を30度ずつ傾けてやるにはどうしましょう．まずひとまわりは360度ですから，グルリとひとまわりするには360÷30＝12回はくりかえさなければいけません．くりかえしは12回，傾きは30度ですから

REPEAT 12 [SHIKAKU120 RT 30]

12回のくりかえし [一辺120の四角を　右へ30度傾けて]

がその基本になります．Logoは1文字分あけることと，文字をつなげることを厳密に区別します．

REPEAT 12とすると，たとえばこれは手続きなのだろうか？ とLogoは考えますし

SHIKAKU 120 とすると，手続きを呼び出すのではなく，なにかの計算ではないか，とLogoは考えて，つじつまが合わないとエラーメッセージを出します．1字分をあける時とそうでないときによく注意します．では，この回転する四角を白（カラーコード7）でしかもゆっくりと描くようにしましょう．それにはSETPC（セット・ペンカラー：色の設定），WAIT（ウェイト：時間待ち）を追加します．？が出ている時にTSにしてから　TO KAITENSHIKAKUとタイプしてリターンキーを押します．

このあと，この回転四角の手続きにSETPCとWAITを追加して，新しい定義をしてください．

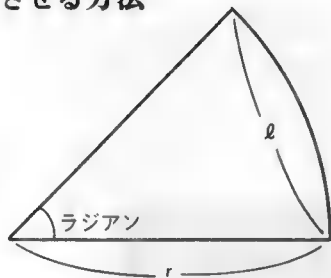
Logoの基本は直線，でもこまかく進むと曲線

＊まがって進むには

Logoのタートルはどんな小さな角度にも向きます．たとえば

REPEAT 100 [RT 0.1 FD 0]

右へ0.1度ずつ，前へは進まないことの100回のくりかえしにも，ちゃんということを聞きます．しかし，ある位置からつぎの位置へと移動するのは，あくまでも直進のみです．プリミティブに円を描かせる命令はないので，自分で作らなければいけません．

＊＊円弧を少しずつ前進させる方法

▲$l=r\times$ラジアン

まず円弧を考えましょう．円弧の長さはこうきまっています．

円弧の長さ＝RADIAN×r　（rは円の半径）

1ラジアンのとき，円弧の長さは半径に等しくなり，πラジアンのとき（3.14159ラジアン）は直径の3.14159倍が円弧の長さになります．πはおなじみの円周率で3.14159，なお，角度との対応は0.5πで90度，πで180度，1.5πが270度，2πは360度です．

いま，ある円弧を描くことにして，とりあえず必要なデータはなんでしょうか？半径(r)と角度(degree)です．まず角度とラジアンの関係を知らないといけません．

360度＝2×π≒6.28　　1度≒6.28÷360（ラジアン）

つまりある半径の円の，ある角度が作る円弧の長さを知りたかったら

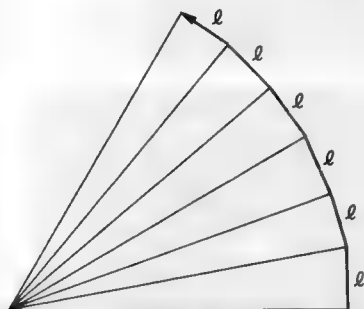
円弧の長さ＝(6.28÷360)×角度×半径

になります．ただし，ここでもとめた円弧の長さはあくまでも計算上のものですから，この長さをもとにしてLogoで

FD 円弧の長さ　　（円弧の長さ分だけ前進）

としても，直線になってしまいます．

▲5°ずつ前進させてやる

そこで，もとめた円弧をある回数ずつ，こきざみに前へ進めます．そして角度もある回数ずつ，こきざみにしなければいけません．たとえば円弧の長さをもとめ，10回に分けて前進させるなら，角度も10に分割するのです．どころで現実にはFM-77のディスプレイ上でこまかく分割しても時間がかかるばかりです．分割する角度は，どんな円弧でも1回につき5度もあれば十分でしょう．角度は5度きざみとしたときは，くりかえし数は「角度÷5」，前へゆく1回分の距離は(6.28÷(360÷5))×r，タートルの首振り角は5度になります．

円弧は円にもなる

＊変数の書きかたに注意

では実際に円弧を描く手続きをしましょう．ここでまず注意することは前へ100進め，というときは“FD□100”と□の部分を1文字あけています．Logoの手続きで，もし前へXだけ進め，と命令し，このXはLogoがべつの場所で計算するものだよ，というときは

FD :X

というように，やはりFDのあとに1文字分をあけ，つぎに：(コロン)を打ち，そのあとにXを打つようにします．また，手続きのタイトルにも，「これからの手続きはXという変数が入ります」ということを知らせるために，やはり1文字分をあけます．もし変数が2つあるときには□：X□：Yというようにしないと，Logoに正しく伝わりません．□：X：Yでは，Logoは“：X：Y”という名前のひとつの変数と思ってしまうのです．《1文字あけが正しくない》というエラーメッセージは出ないので，十分に注意してください．

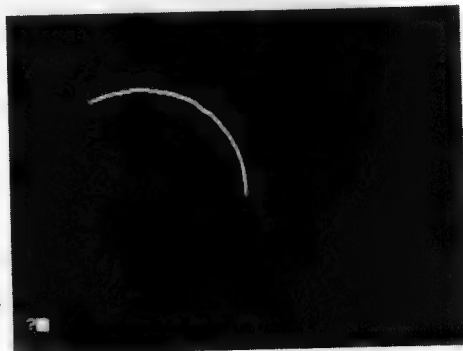
```
TO Lｴﾝｺ :R :DEG
JUNBI
SETPC 7
HT
REPEAT :DEG / 5 [FD :R * 0.0872 LT 5]
END

TO Rｴﾝｺ :R :DEG
JUNBI
SETPC 7
HT
REPEAT :DEG / 5 [FD :R * 0.0872 RT 5]
END

TO JUNBI
HOSEI
HT
SS
CS
END

TO HOSEI
.SETSCRUNCH 0.435
END
```

▲右円弧，左円弧の手続き

◀右まわり

左まわり▶

**手続きを終えたらSAVEしよう

いままでの話で，まだよく分らない人は，とにかく一度電気を切ってしまいましょう．そしてディスケットに書かれたファイルの

ERF "D0：HOSEI　（ERF：イレースファイル）

ERF "D0：JUNBI

の2つを消してしまって，前ページの4つの手続きを改めて定義してもかまいません．前ページの手続きがすべて済んだら，Lエンコ（左むきの円弧），Rエンコ（右むきの円弧）を何回か，試してみましょう．どちらもまちがいなくできることを確認したら

SAVE "D0：エンコ

で，このプログラムをディスケットにおさめましょう．ねんのため，このあとに

ERPS　（すべてのプロシジャをワークスペースから消去）

とタイプし，リターンキーを押したあと

POTS　（画面に，いまあるプロシジャを出しなさい）

を実行しましょう．画面にはひとつのプロシジャもたくわえられていないため，？が出るだけのはずです．そうして

LOAD "D0：エンコ

と，ディスケットから手続きを読みこみます．JUNBIの手続きは前のものとは違う順序ですが，結果はほとんど同じです．

***つぎのステップのための応用

Logoがなにをしているのか，分ってきましたか？いまの円弧ではもっとも原始的な手続きが「補正」で，「準備」にはその補正といくつかのプリミティブが集まり，円弧を描く手続きではそれに加えて，白い色を指定したり，準備の手続きを呼んだりしています．

なおこの円弧を描くときの，BASICでいえば"RUN"に当るものは，もし右まわりの円弧で半径が120，角度が100度なら？のあと

?Rエンコ　120　100

としてリターンキーを押します．

連続して右まわり，左まわりをさせるときは「準備」の手続きなどを工夫してみましょう．なおPU（ペンアップ）でタートルを好きなところへ移動できますが，描く前にはかならずPD（ペンダウン）しないと，線は描かず，タートルだけがチョコチョコ動いてしまいますので注意しましょう．円は角度を360にすればいいですね．

連続して動かす手続きも，別の名前でSAVEするとよいでしょう．

花壇には花が，花にはなにが？

＊部分から全体を作る

花壇にはなにがありますか？花があります．１本の花にはなにがありますか？花びらと葉と茎があります．

ではLogoで花壇を作るにはどうしたらよいでしょうか．いま述べた考えのとおりにそれぞれの部分を作り，組合せればよいのです．それぞれの手続きを定義して，さいごのメインとなる手続きで完成させます．まず花びらを作りましょう．花びらは右まわりの円弧と，左まわりの円弧を組合せればよいでしょう．ここでは“HANABIRA”の手続きで作っています．この手続きの中の

RT 15 × 6

は右まわりの円弧を描いたあとのタートルの向きを変えるためです．

花びらがいくつか集まると花になります．“HANA”の手続きで，花びらを８枚描くようにしました．それぞれの花びらは45度ずつ，向きを変えています．葉も花びらとおなじように“ハ”で手続きをし，これを茎のどこかにつけてやればよいでしょう．これらの手続きを組合せると，１本の花ができます．

まず茎です．“クキ”は

まず30だけ進み→15度分タートルをかたむけ→葉を描き→タートルを０度にもどし→40進み……というように，茎と葉との組合せになっています．１本の花は，さらにこの茎の上に花が乗りますから，これを“ハナ”という名前で手続きをしました．花壇ですから，何本かの花を植えなければいけません．タートルを動かし，線を描かないようにするためのプリミティブも使います．

※手続きは，作る順序は関係ない．掲載されたものを順不同で打ってもよい

```
TO ｶﾀﾞﾝ
HOSEI
ｻｶﾞﾙ
PU
LT 90
FD 240
SETH 0
PD
REPEAT 4 [PD ﾊﾅ ｼﾀﾍﾓﾄﾞﾙ PU RT 90 FD 160]
END
```

▲花壇の手続き

花壇のできあがり▶

**いままでのおさらい

ではいちど，ここでLogoのおさらいをしましょう．まず手続きです．手続きは，Logoにさせたいことを教えることです．手続きには名前をつけてやり　TO　手続きの名前　という形で定義します．手続きの終了はリターンキーではなく，ESCキーを使います．Logoにはプリミティブと呼ばれる命令があり，手続きをしなくても直接使えます．ただし，英語で“I AM”を“IAM”と書くとまったく別のものになるように，Logoでは“FD　100”と“FD100”を区別します．“FD 100”なら《100だけ前進》と理解しますが，“FD100”ではそれに対応するものをLogoが探し出し，あなたが“FD100”をきちんと教えていないと，エラーメッセージを出します．いままで出てきたプリミティブをまとめましょう．

PU：ペンアップ（タートルは線を描きません）
PD：ペンダウン（タートルは線を描きます）
SETH：タートルの向きを何度にするかを命令します．0では上向き，90では時計まわりに90度を向きます．
REPEAT：[　] の部分を，指定の回数くりかえします
RT：タートルを右向きにします
LT：タートルを左向きにします
FD：前進の命令です
BK：後退の命令です．タートルの向きは変えません
SETPC：色を指定します
SS：スプリットスクリーンで，タートルが活躍する場所を確保します．画面がこわれることが少なくなります．
TS：テキストスクリーンで，タートルを使わないとき便利です．
.SETSCRUNCH：タテの比率を指定します．さいしょにピリオドがつくことに注意しましょう
SAVE “DO：ワークスペースにあるものを，ディスケットに納めます
LOAD “DO：ディスケットに納められた手続きなどをLogoに伝えます
POTS：ワークスペースにある手続きを画面に出します
POPS：ワークスペースにある手続きの内容を画面に出します
ERPS：ワークスペースにある手続きをすべて消します
ER [　]：指定した手続きを消します

```
TO ﾊ
REPEAT 15 [FD 4 RT 6]
RT 15 * 6
REPEAT 15 [FD 4 RT 6]
END

TO ｸｷ
SETPC 4
SETH 0
FD 30
SETH 15
ﾊ
SETH 0
FD 40
SETH 280
ﾊ
SETH 0
FD 130
END

TO ﾊﾅ
ｸｷ
SETPC 2
HANA
END

TO ｼﾀﾍﾓﾄﾞﾙ
SETH 0
PU
BACK 200
PD
END

TO HANABIRA
REPEAT 15 [RT 6 FD 6]
RT 15 * 6
REPEAT 15 [RT 6 FD 6]
END

TO HANA
SETPC 3
REPEAT 8 [HANABIRA RT 45]
END

TO HOSEI
.SETSCRUNCH 0.435
END

TO ｻｶﾞﾙ
PU
BACK 120
PD
END
```

▲その他の8つの手続き

タートルのまとめに，きれいな花たばを

*コツはタートルの向きが上になること

いままでのLogoによるタートルグラフィックスは，できるだけ多くのプリミティブ（Logoに備わっている命令）を紹介することもあって読んでいるあなたには整理がつかなかったかもしれません．タートルグラフィックスのしめくくりに，少しの手続きで作れる美しい花を描きましょう．テーマは基本的に同じものです．

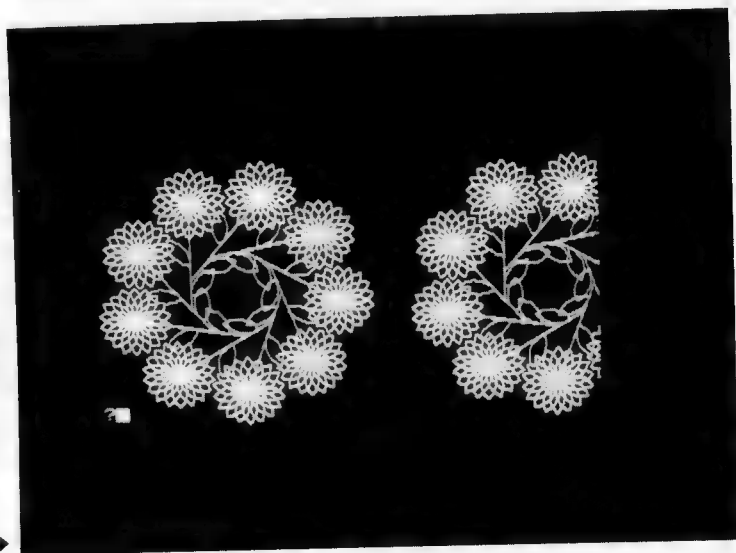
まず，“ハナビラR”を見てください．ここでは

```
REPEAT 10 [FD :SIZE RT 9]
```

（サイズ分進み，そのあと9度向きを変える円弧を10回描く）

ことにしています．9度の向きを10回ですからちょうど90度だけタートルはその向きを変えます．そして“RT 90”で合計180度向きを変え，さらに花びらの下半分を描きます．描いたあとでは270度向きが変わり，さらに“RT 90”で総合計が360度になります．花びらをひとつ描くとタートルは，またさいしょの向きになっていますから，そのあとの処理がラクになるのです．ちなみに“ハナ”の手続きを見てください．

ここでも“リピート18，右まわり20度”ですから，やはり360度になっています．花を描き終ったときもタートルは最初の向きです．

※TSはテキストスクリーン

ごくわずかの手続きでこんな花が▶

**主手続きにだけインプットがあればよい

このような処理を使えば，花全体を回転させても，つねに中心からずれてゆくことはありません．“ハナビラR”などは，葉を描くためにも使っていますが，１枚の葉を描いてそのままで前進させて茎を描いても，１本の花全体はまっすぐになる，というわけです．

なお，この回転をする花では主手続きが“カイテンハナ　:SIZE”になっていますから，？が画面に出たとき（これをトップレベルといいます）に

カイテンハナ　○○（例：カイテンハナ　4）

として，○○に寸法を入れます．このように手続き名のあとにさらに数値などを入れる（インプット）と，Logoは他の手続きを呼び出すときにも　:SIZEをおぼえているので，その他の手続きでいちいちインプットする必要はありません．ただし，たとえば“クキ”だけを直接呼び出しても，Logoは動かないことは分りますね．

※SETX，SETYはタートルを絶対座標上に動かすプリミティブ

```
TO ｶｲﾃﾝﾊﾅ :SIZE
TS
HOSEI
PU
SETY -50
SETX :SIZE * ( -40 )
PD
REPEAT 9 [ﾊﾅ RT 40]
PU
SETX :SIZE * ( 40 )
PD
REPEAT 9 [ﾊﾅ RT 40]
END

TO ﾊﾅ
SETPC 4
ｸｷ
SETPC 6
REPEAT 18 [ﾊﾅﾋﾞﾗR RT 20]
SETPC 4
BK :SIZE * 28
END

TO HOSEI
.SETSCRUNCH 0.435
END
```

```
TO ﾊﾅﾋﾞﾗR
REPEAT 10 [FD :SIZE RT 9]
RT 90
REPEAT 10 [FD :SIZE RT 9]
RT 90
END

TO ｸｷ
FD :SIZE * 4
ﾊﾅﾋﾞﾗR
FD :SIZE * 8
ﾊﾅﾋﾞﾗL
FD :SIZE * 16
END

TO ﾊﾅﾋﾞﾗL
REPEAT 10 [FD :SIZE LT 9]
LT 90
REPEAT 10 [FD :SIZE LT 9]
LT 90
END
```

▲回転する花の手続き

好きな絵を作る“シェイプ”

＊シェイプの作りかたとディスケットへのセーブ

Logoのシェイプ（shape：形）コマンドを使うと，好きな形のパターンを作ることができます．その基本は

1．ESコマンドでシェイプ・エディタを呼び出す．たとえば星条旗(アメリカの国旗)を作り，そのシェイプに“USA”という名前をつけたかったら，トップレベルで

?ES "USA

としてリターンする．

2．画面に現われたパターンを，テンキー，カーソルキーその他を使って編集する．終ったらESCキーを押す．

3．このパターンは見かけ上，タートルとして使えるので，もしデフォルト時のタートル（三角マーク）の代りに使いたかったら

?SETSH "USA

などとすればよい．

ということになります．

```
TO ハタ
SETSH "USA1
STAMP
RT 90 FD 32
SETSH "USA2
STAMP
LT 90
END
```

▲星条旗

左のサンプルは星条旗を左右に分けて，2つのシェイプとして作りそれを画面の中央に出力するプロシジャです．“USA1”が星のある部分，“USA2”は赤と白のストライプ部分です．実際には

?ES "USA1

で左半分を作ったらESCキーを押し，

?ES "USA2

で新しいシェイプを作ったら同じようにESCキーを押します．これでワークスペースには2つのシェイプが登録されます．そのあとに左の手続きを作成し，たとえば

SAVE "D0：USAFLAG

などとすれば，このファイルには手続きと共に2つのシェイプに関するパターンデータも一緒にセーブされます．

＊＊スタンプ命令で好きな場所に出力

このシェイプは単にタートルの代りになるだけでなく，スタンプ命令その他によって画面に自由に出すことができます．次のページは画面に星条旗をたくさん出力し，最後には元の4倍の大きさのものをひとつだけ出力するものです．

手続きは全部で4つあります．ひとつは旗を1個だけ出力するもので，シェイプはタテに16ドット，ヨコに32ドットが基本になっているために“ハタ”の手続きでは左半分をスタンプしたら，右へ32ドット分ずれて残りの右半分をスタンプします．“イチレツノハタ”は横一列に9個の旗を出力します．“タクサンノハタ”は，タートルを左上に設定し，一列の旗を上下に7回出力する手続きです．一列の旗を1回出力すると32(ドット)＋40(ドット……余白分)の9倍だけ右へ移動するので，(32＋40)×9ドット分，左へ戻しています．

小さい旗を63個出力したら，タートルはまた画面の左上へと移動して，さいごに“オオキイハタ”を呼び出しています．さいごにタートルを消している（HT）ことに注意してください．

※下のプログラムは“タクサンノハタ”を入力すると実行される

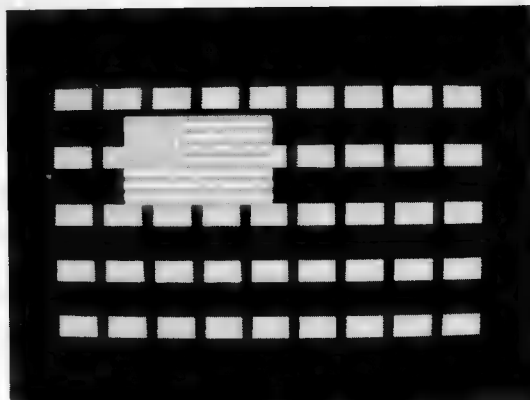
```
TO ﾀｸｻﾝﾉﾊﾀ
PU CS HOME
HT LT 90 FD 300 RT 90 FD 300 ST
REPEAT 7 [ｲﾁﾚﾂﾉﾊﾀ LT 90 FD ( 32 + 40 ) * 9 LT 90 FD 90 SETH 0]
SETH 90 FD 150 SETH 0 FD 350
ｵｵｷｲﾊﾀ
HT
END

TO ｵｵｷｲﾊﾀ
SETSH "USA1
BIGSTAMP 4 4
RT 90 FD 32 * 4
SETSH "USA2
BIGSTAMP 4 4
LT 90
END

TO ｲﾁﾚﾂﾉﾊﾀ
REPEAT 9 [ﾊﾀ RT 90 FD 40 LT 90]
END

TO ﾊﾀ
SETSH "USA1
BIGSTAMP 1 1
RT 90 FD 32
SETSH "USA2
BIGSTAMP 1 1
LT 90
END
```

実行例▶

▲星条旗をつぎつぎと画面に出す

Logoのもうひとつの世界への第1歩

＊MAKEというプリミティブ

このへんでLogoのグラフィックスから離れて，べつの面を見てゆきましょう．BASICになじんでしまうと，Logoはタートルグラフィックスは別として，つきあいにくい，という人も多いでしょう．考えかたは厳密で文法がこまかく規定されていないので，ルールがつかみにくい，変数や変数名についても自由度が大きい代りに，使いかたがむずかしいともいえます．

いずれにしても少しずつ，Logoの使いかたをマスターするしかありません．手はじめに，2つの数を扱うことから考えてゆきましょう．内容はごく簡単ですが，書法をマスターする，といううえではちょうどよいものかもしれません．ここには2つの手続きがあります．もともとは，2つにする意味はないのですが，分けることはあなたがこの先Logoによる長いプログラムを作るときに役に立つでしょう．“ダイショウ”という手続きは，ほとんどが説明文です．

PR [　　]　　　(PRはPRINTの略)

は [　] の中の文章を画面に出します．なお，Logoはパーシングといって，使い手がタイプしたものをLogoにつごうのよいように内部で処理します．その処理のひとつに文字と文字は2文字以上の空白は作ら

※SE：センテンスの略
※PR：プリントの略
※PR [　]は結果的には1行あけになる

```
TO ﾀﾞｲｼｮｳ
PR [ｺﾚｶﾗ 2ﾂﾉ ｶｽﾞﾉ ﾀﾞｲｼｮｳ ｦ Logo ﾆ ﾊﾝﾃｲｻｾﾏｽ]
PR [ ]
PR [ｶｽﾞｦ ﾋﾄﾂ ｽﾞﾂ ﾆｭｳﾘｮｸ ｼ RETURN KEY ｦｵｼﾃｸﾀﾞｻｲ]
PR [ ]
ﾋｶｸ
END

TO ﾋｶｸ
PRINT [ﾊｼﾞﾒﾉ ｶｽﾞ ?]
MAKE "A FIRST READLIST
PRINT [ﾂｷﾞﾉｶｽﾞ ?]
MAKE "B FIRST READLIST
IF :A < :B [PRINT ( SE :A [<] :B )]
IF :A > :B [PRINT ( SE :A [>] :B )]
IF :A = :B [PRINT ( SE :A [=] :B )]
PRINT [ ]
ﾋｶｸ
END
```

2つの数の比較▶

ないものがあり，あなたがどんなにがんばってもカズノ□□□ダイショウというように3文字分をあけることはできません.

“ヒカク”では，Aという変数に，キーボードから入力した数値をあてはめる作業があります．それがMAKEです.

```
MAKE "A FIRST READLIST
```

BASICの「LET A」と「INPUT」の組合せに似ています．リードリストがキーからのリスト入力を意味します．なお，ファーストを使わずに

```
MAKE "A READLIST
```

とすると，LogoはAという変数にリストをあてはめます．結果的にはAには数値が代入されません．ファーストリードリスト……つまりリストの中の，さいしょの要素をAに入れる，ということで，はじめて正しい関係になります.

▲リストの考えかた

リストという言葉は，BASICではもっぱらプログラムリストという意味で使われますがLogoのリストはさいしょとさいごが［ ］でかこまれたもの，と解釈します.

［山 川 ［山と川］ 海 スキー場］

というリストは，ぜんぶで5つの要素のリスト，とLogoは考えます.

**IFとその処理の考えかた

IFの処理の基本はこうです.

IF 条件 ［処理1］ ［処理2］

条件が真なら処理の1を，偽なら処理2を実行します．“ヒカク”の手続きでは，2つの数の関係を，大きいか，小さいか，等しいかの3つに分けて処理するので，ここでも正直に3つの判断をさせました．したがって，この手続きには［処理2］は書かれていません．なお，処理の内容はPRINTすることです．そして，PRINTの内容はセンテンス（文章）です．このセンテンスは変数や文字を含んでいますから，PRINT（　　）という形にしています．［＝］とするのはなぜか，についてはもう分るでしょう.

このプログラムは？が画面に出ているときに

ダイショウ

とタイプしてリターンキーを押して始めます．“ヒカク”の手続きでさいごに“ヒカク”自身を呼んでいるため，永久に終りません，やめるときはBREAKキーを押します.

公約数と公倍数を計算させましょう

*ユークリッドの互除法

小学校の算数で，私たちは最大公約数と最小公倍数というものを習います．ちょっとした数なら経験からすぐに答がみつかります．しかし，たとえば4ケタや5ケタの数ともなると計算しなければいけませんね．Logoにさせましょう．ところで，これをもとめるには《ユークリッドの互除法》というものがあります．とりあえず2つの数ということにしましょう．

1．はじめの数のうち，大きい数÷小さい数　の余りをもとめる
2．余りがあったら，つぎにその余りと1.の作業で得た小さい数とを比べる.そしてやはり　大きい数÷小さい数 の余りをもとめる
3．この作業を余りがなくなるまでくりかえす
4．余りがなくなったときの割る数が最大公約数になり，最小公倍数は，はじめの数 ×（もうひとつのはじめの数÷最大公約数）ということになる

というのがそれです．たとえば数Aが16，Bが72としましょう．

1．A＝72　B＝16におきかえて，72÷16の余りをもとめる（答は商が4，余りが8）．余りをAとし，Bは16
2．A＝16　B＝8におきかえて　16÷8の余りをもとめる．答は2で余りが0なので，これで終り
3．最大公約数はいちばんさいごのBなので8．最小公倍数は（72×(16÷8)＝144になる(16×(72÷8) でも同じです)．

**変数をたがいに交換する

Logoでこれらの手続きをするためには，まずいくつかの手法を考えなければいけません．余りをもとめるのは，BASICにはMODがあったようにFM LogoにはREMAINDER（リメインダ）があるのでそれを使います．もうひとつの，変数の中味の交換ですが，これもF-BASICにはSWAPがありますが FM Logo にはないので，工夫が必要です．

2つの変数はそれぞれ2つのコップに入ったものとして，それらを同時に交換できない（SWAPではそれができます）ときはどうしますか？もうひとつのコップを用意して，まずAの中味をこのスペアのコップに入れ，つぎにBの中味をAにあけます．そしてBにスペアのコ

```
TO SWAP :A :B
MAKE "A :B
PR :A
PR [ ]
MAKE "B :A
PR :B
END
```

▲この手続きでは変数は交換できません

ップの中味を入れればよいのとおなじ手法を使いましょう．これが“ソウサ”の手続きの中にある

MAKE "X :A MAKE "A :B MAKE "B :X

です．スペアのコップとおなじはたらきを，変数Xにさせています．

ユークリッドの互除法をLogoにさせるのはそれほどむずかしくはありません．“ソウサ”の手続きの中で“ソウサ”を呼ぶという，再帰的な手法を使うわけですが，ある特定の条件のときにこの再帰をやめさせることを考えないと，永久に続いてしまいます．

数の大小をくらべて入れかえる
余りをもとめる
数の大小をくらべて入れかえる
余りをもとめる

とくりかえされるループの中の，どこかに判断をさせる手続きを書いてやるわけです．ここに掲載した手続きは，ごく原始的で洗練度の低いものですが，タートルグラフィックス以外のLogoを理解するための初歩的な手続きということで，あえて作っています．

主手続きは“コウヤク”です．80桁にする手続き，入力手続き，操作手続きで構成されています．テキストスクリーン（TS）を使うことを命令したあと

?コウヤク

とタイプし，リターンキーを押したあとは画面にしたがって操作できます．なお，入力した数値がマイナスのときは，Logoは永久に堂々めぐりをします．また，これらの手続きをプリンタに出力するには

SAVE "P:

というプリミティブを使います．ワークスペースにあるもっとも新しい変数のすべてもプリンタに出力されるのがおもしろいですね．

```
TO ｺｳﾔｸ
80CHAR
INPUT
ｿｳｻ
END
```

▲主手続き

```
TO INPUT
PRINT [ﾊｼﾞﾒﾉ ｶｽﾞ ?]
MAKE "A FIRST READLIST
PRINT [ﾂｷﾞﾉｶｽﾞ ?]
MAKE "B FIRST READLIST
MAKE "ﾊｼﾞﾒﾉA :A
MAKE "ﾊｼﾞﾒﾉB :B
END
```

▲入力の手続き

```
MAKE "B 3
MAKE "A 0
MAKE "X 3
MAKE "ﾊｼﾞﾒﾉB 15
MAKE "ﾊｼﾞﾒﾉA 12
```

▲ここで使われる変数の例

```
TO ｿｳｻ
IF :A = 0 [PRINT ( SE [ｻｲﾀﾞｲ ｺｳﾔｸｽｳ =] :B [ｻｲｼｮｳ ｺｳﾊﾞｲｽｳ =] :ﾊｼﾞﾒﾉA * ( :ﾊｼﾞﾒﾉB
 / :B )]
IF :A = 0 [STOP]
IF :A < :B [MAKE "X :A MAKE "A :B MAKE "B :X]
MAKE "A REMAINDER :A :B
ｿｳｻ
END
```

▲操作の手続き

判断をして，手続きを呼び出す

＊このような手続きも可能です

Logoの使いかたのもうひとつとして，判断を述べたあとの処理（プレディケイトと呼びます）に，手続きを呼び出す方法を紹介します．前のページに出ていた，入力した数値の大小の比較と同じような考えかたですが，基本的にはまったく異なる手法です．数値はまず，ゼロかそうでないかでふるい分け（“ニュウリョク”の手続き）て，ゼロなら“ゼロ”の，そうでないのなら“ゼロデハナイ”手続きを呼びます．“ゼロデハナイ”手続きでは，プラスかマイナスかの判断をしています．[マイナス　ニュウリョク]は，“一の入力”という意味ではなく，“マイナス”の手続きを呼び，つぎに“ニュウリョク”の手続きを呼ぶことです．なお手続きは何重にも呼び出せます．

※？テツヅキから始めます
※RL：リードリストの略
※PR：プリントの略

```
TO` ﾃﾂﾂﾞｷ
PR [ｺﾉ ﾐﾎﾝﾊ ﾊﾝﾀﾞﾝｦｼﾃ ﾃﾂﾂﾞｷｦ ﾖﾌﾞﾓﾉﾃﾞｽ]
PR [ ]
PR [ｷｰﾎﾞｰﾄﾞｶﾗ ｽｷﾅｶｽﾞｦ ｲﾚﾃｸﾀﾞｻｲ]
ﾆｭｳﾘｮｸ
END

TO ｾﾞﾛﾃﾞﾊﾅｲ
IF :ｶｽﾞ < 0 [ﾏｲﾅｽ ﾆｭｳﾘｮｸ] [ﾌﾟﾗｽ ﾆｭｳﾘｮｸ]
END

TO ｾﾞﾛ
PR [ｶｽﾞﾊ ｾﾞﾛﾃﾞｽ THE END!!!!]
END

TO ﾆｭｳﾘｮｸ
PR [ｻｱﾄﾞｳｿﾞ RETURN KEY ｦ ｵｽﾉｦ ｵﾜｽﾚﾅｸ]
MAKE "ｶｽﾞ FIRST RL
IF :ｶｽﾞ = 0 [ｾﾞﾛ STOP] [ｾﾞﾛﾃﾞﾊﾅｲ]
END

TO ﾏｲﾅｽ
PR [ｿﾚﾊ ﾏｲﾅｽﾃﾞｽ ﾆｭｳﾘｮｸﾍ ﾓﾄﾞﾘﾏｽ]
END

TO ﾌﾟﾗｽ
PR [ｿﾚﾊ ﾌﾟﾗｽﾃﾞｽ ﾏﾀ ﾆｭｳﾘｮｸﾍ ﾓﾄﾞﾘﾏｽ]
END
```

▲判断をしたのち，手続きを重ねて呼ぶ例

手続き
前処理
入力
カズ←KEY IN
数＝0？
偽
真
ゼロ
STOP
ゼロではない
数は－？
偽
真
プラス
マイナス

▲プログラムの考えかた

ワードとリストをしっかり理解しましょう

＊リストはノート，ワードは言葉

Logoが文字やことばを扱うとき，私たちはLogoがどう処理しているかをよく理解し，Logoに合せて手続きを書かなければいけません．もういちど《リスト》の考えをしっかりつかんでおきましょう．

MAKE “A RL （リストをキーボードから入力し，Aに代入)の手続きは，じっさいにあなたがキーボードから

ハト　マメ　マス

を打ちこんで，リターンキーを押すと［ハト　マメ　マス］がAという変数名のついた記憶の箱に入れられます．

ハト

だけなら，［ハト］がAという変数になります．

MAKE “A FIRST RL

の手続きは，キーボードから入力したリストのうちの，さいしょのことばをAに代入するものですから，Aは，たとえば“ハトになります．

［ハト］というリストと“ハト　というワードはまったくちがうものであることに気付いてください．［　］はリストで，ちょうどノートにひとつのことばが書かれていると［ハト］になり，いくつかのことばが書かれていると［ハト　マメ　マス］になります．ワードはあくまでもワードですから，ノートを必要としません．

下に示したサンプルは変数を3つ用意して，そこに何が入っているかを見るものです．リストという変数はリストをあつかい，他のふたつは文字をあつかっている（プリントアウトしたものの［　］と“に注目してください)．FIRST FIRST :A は「Aという変数名にある，さいしょのことばのさいしょのもの」ということです．

※？リストでプログラムを実行

```
TO ﾘｽﾄ
MAKE "ﾘｽﾄ RL
MAKE "ﾘｽﾄﾉﾊｼﾞﾒﾉｺﾄﾊﾞﾉﾊｼﾞﾒﾉﾓｼﾞ FIRST FIRST :ﾘｽﾄ
MAKE "ﾘｽﾄﾉﾊｼﾞﾒﾉｺﾄﾊﾞﾉｻｲｺﾞﾉﾓｼﾞ LAST FIRST :ﾘｽﾄ
END

MAKE "ﾘｽﾄ [ﾎﾝｼﾞﾂﾊ ｾｲﾃﾝ ﾅﾘ]
MAKE "ﾘｽﾄﾉﾊｼﾞﾒﾉｺﾄﾊﾞﾉｻｲｺﾞﾉﾓｼﾞ "ﾊ
MAKE "ﾘｽﾄﾉﾊｼﾞﾒﾉｺﾄﾊﾞﾉﾊｼﾞﾒﾉﾓｼﾞ "ﾎ
```

※下の3つのMAKEは，SAVE “P：を実行したときの，ワークスペースにある変数とその内容を表わしている

◀リストを入力し，文字をとり出す

Logoに“しりとり”の審判をさせる

＊しりとりのルール

リストとことば（あるいは文字）をよく理解する見本として，Logoによる“しりとり”ゲームを考えましょう．しりとりのルールは

○ことばの終りが“ン”ならゲーム不成立

○前のことばとつぎのことばが同じ文字でなければゲーム不成立

○いちど出たことばをもう一度出したらゲーム不成立

ということです．ここではカタカナを使いますが，たとえば日本語の“ガ”は，Logoはひとつの文字としてみとめず，さいごの文字は“ﾞ”に扱いますが，この点については今回は考えません．

＊＊ことばを入力し，リストを作る

いまのルールを，そのままLogoに教えこめばよいので，それほどむずかしくはないでしょう．ここでは：Aにはじめのことば，：Bはつぎのことば，としてやります．入力としては

：A→：B→(：Bを：Aに変える)→：B→(：Bを：Aに変える)……

ということにし，これをくりかえしてやればよいのです．“はじめの入力”の手続きで

MAKE “A FIRST RL

とし，：Aにはことばを代入してやり

MAKE “デタコトバ　LPUT ：A ［ ］

で，出てきたことばをリストのうしろにつけてやります．

※LPUT：リストのさいごにリストやことばを加える

```
TO ｼﾘﾄﾘ
40CHAR
TS
PR [ ]
PR [ ]
PR [ｺﾚｶﾗ ｶﾀｶﾅﾆﾖﾙ ｼﾘﾄﾘ ｦ ｼﾏｼｮｳ]
PR [ ]
PR [ ]
PR [ｺﾄﾊﾞﾉ ﾊｼﾞﾒﾔ ｵﾜﾘﾉ]
PR [{ﾀﾞ} ﾔ {ﾊﾟ} ﾔ {ｯ} ﾅﾄﾞﾊ {ﾀ ﾊ ﾂ} ﾄｼﾏｽ]
PR [ ]
PR [ﾓﾁﾛﾝ {ﾝ} ｶﾞ ﾃﾞﾀﾗ ｵﾜﾘﾆ ﾅﾘﾏｽ]
PR [ｲﾁﾄﾞ ﾃﾞﾀ ｺﾄﾊﾞ ﾊ ﾓﾁﾛﾝ Logo ｶﾞ ｵﾎﾞｴﾃｲﾏｽ]
ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ
END
```

▲この手続きからゲーム開始

```
TO ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ
WAIT 500
CLEARSCREEN
SPLITSCREEN
HIDETURTLE
PR [ｻｱ ｼﾘﾄﾘ ｦ ﾊｼﾞﾒﾏｼｮｳ]
PR [ ]
PR [ｺﾄﾊﾞｦ ﾄﾞｳｿﾞ]
MAKE "A FIRST RL
IF "ﾝ = LAST :A [ﾝﾃﾞｵﾜﾘ ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ]
MAKE "ﾃﾞﾀｺﾄﾊﾞ LPUT :A [ ]
ﾂｷﾞﾉｺﾄﾊﾞ
```

▲はじめの入力の手続き

```
TO ﾝﾃﾞｵﾜﾘ
PR [ ]
PR [ﾝ ﾃﾞｵﾜｯﾃｲﾏｽ ｹﾞｰﾑｾｯﾄ]
ｶｳﾝﾄ
END

TO ﾓｳﾃﾞﾏｼﾀ
PR [ ]
PR [ｿﾉｺﾄﾊﾞﾊ ﾓｳﾃﾞﾏｼﾀ]
ｶｳﾝﾄ
END

TO ﾂﾅｶﾞﾘﾏｾﾝ
PR [ ]
PR [ｼﾘﾄﾘﾆ ﾅｯﾃｲﾏｾﾝ]
ｶｳﾝﾄ
END
```

▲ゲームセットになるときの３つの手続き

```
TO ﾂｷﾞﾉｺﾄﾊﾞ
PR [ﾂｷﾞﾉｺﾄﾊﾞ ｦ ﾄﾞｳｿﾞ]
MAKE "B FIRST RL
MAKE "AEND LAST :A
MAKE "BFIRST FIRST :B
TEST :AEND = :BFIRST
IFFALSE [ﾂﾅｶﾞﾘﾏｾﾝ ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ]
IF "ﾝ = LAST :B [ﾝﾃﾞｵﾜﾘ ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ]
IF MEMBERP :B :ﾃﾞﾀｺﾄﾊﾞ [ﾓｳﾃﾞﾏｼﾀ ﾊｼﾞﾒﾉﾆｭｳﾘｮｸ]
MAKE "ﾃﾞﾀｺﾄﾊﾞ LPUT :B :ﾃﾞﾀｺﾄﾊﾞ
MAKE "A :B
ﾂｷﾞﾉｺﾄﾊﾞ
END
```

▲これが実際的にはメインの手続きになります

つまり：Aと：Bはことば(ワード)で，：**デタコトバ**はリストになります．“つぎのことば”が主要手続き，ともいうのもので，ここでは：Bの入力をもとめたあと，：Aのことばと：Bのことばのさいごとさいしょがつながっているか，“ン”で終ることばか，いままでに出たことばか（MEMBERP）どうか，などのゲームの成立要素をチェックしています．

そして，それらがまったく問題ないときは，次のことばの入力のために，いままで出たことばの一覧表を作ります．

MAKE “デタコトバ　LPUT　：B　：デタコトバ

(：デタコトバというリストのさいごに，：Bという変数のことばをつけて，新しく：デタコトバというリストを作りなさい)

がそれです．そしてもうひとつ，：Aに：Bを代入して，またゲームを続けます．

SAVE　“P：で手続きと共にプリンタに打出される変数などを見ると，ことばとリストと変数名の関係がはっきりするでしょう．サンプルは“リボン”を入力したためにゲームセットとなったときです．

※このしりとりは７つの手続きから構成されている

カウントの手続き▶

```
TO ｶｳﾝﾄ
PR ( SE [ﾃﾞﾀｺﾄﾊﾞﾉｶｽﾞ ﾊ] COUNT :ﾃﾞﾀｺﾄﾊﾞ [ﾃﾞｼﾀ] )
END
```

ゲームセットになったときのワークスペースの中味▶

```
MAKE "ﾃﾞﾀｺﾄﾊﾞ [ｲｽ ｽｲｶ ｶｻ ｻﾒ ﾒﾀﾞｶ ｶｲ ｲﾉﾁ ﾁｷｭｳ ｳｼ ｼﾀｼﾞｷ ｷﾂﾈ ﾈｺ ｺﾏ ﾏﾘ]
MAKE "B "ﾘﾎﾞﾝ
MAKE "A "ﾏﾘ
MAKE "BFIRST "ﾘ
MAKE "AEND "ﾘ
```

名簿を整理し，重複した名前を取りのぞく

＊名簿上に重複した名前があるとき

電話帳でも，名刺のホルダーでもなんでもよいのですが，そこに書き入れたり，しまっておいたものが，重複することがあります．重複したものをとりのぞくことをLogoに命令しましょう．

たとえばいま，ある紙にアルファベットをどんどん書いてゆきます．これがリストの作成ですが，たとえば

A B D F B G D A B

になったとしましょうか．このリストから重複しているものを見つけ出して，新しいリストに

A B D F G

の5つが作成されれば，重複した分をとりのぞいたことになります．どういう手法でプログラムを作りますか？

＊＊重複したものを取りのぞく方法

たとえばこのような方法が考えられます．まず，もとのリストからさいしょの要素を取出します．

A [B D F B G D A B]

そして，取出したAが，右側のリストに含まれているかどうかをチェックします．ここでは右側にAがありますから，新しいリスト（さいしょはなにも書かれていない，まっしろなリストです）には，いま取出した要素は書かないことにします．そして，リストはいま左側に取出した要素を捨ててしまい

B D F B G D A B

というものにします．Bを取出すとここでもBは右側のリストの中に含まれているので，新しいリストには書きこみません．重複していないものだけを書きこみます．このくりかえしで，もとのリストに調べる要素がなくなると，作業を終えます．

※要素が含まれるかどうかのためにはMEMBERPが使える

＊＊＊手続きの作成と注意するところは？

Logoへの手続きも，いまの考えをそっくりはてはめればよいことになります．ただし，くりかえし――再帰的な手続きのどこかに条件判断を入れないと，Logoの実行は永久に終りません．この部分さえじょうずに書けば，あとは特別な手法は不要です．では実際の手続きを見てください．主手続きは“名簿の整理”です．メッセージを画面に出

し，空白のリストを3つ作り，そのあとに“入力”“チェック”そして“出力”の手続きを呼んでいます．

入力は“一人分”という変数にRL（リードリスト）を代入します．ここではひとりひとりを入力し，もし作業を中止するのなら9999をタイプするようにしました．最初の例のように入力すると，“一覧”と名付けたリストは

A B D F B G D A B 9999

になります．この9999は，この先の“チェック”にも使います．なおSTOPというプリミティブはすべての手続きを中断するのではなくそれが書かれている手続きの実行だけを中断します．

チェックは“一覧”のリストからひとつを取出し，これを“サンプル”という変数にします．そして新しくサンプルを取りのぞいた“一覧”を作り，この2つを比べるわけです．サンプルが一覧に見あたらなければ，“新一覧”にサンプルを書きこみ，見あたればそれは重複分ですから“重複”リストに書きこむようにしました．なお一覧リストに9999が残った時点で作業を終えます．プリンタに出力された変数を見てください．重複したものと新しい一覧表が出力されています．

※取出した要素が重複していないとき，その要素をもとのリストのさいごにつける方法もある．その方が合理的といえる

```
TO ﾒｲﾎﾞﾉｾｲﾘ
PR [ﾅﾏｴ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ]
PR [ﾆｭｳﾘｮｸﾊ ﾋﾄﾘｽﾞﾂﾃﾞｽ]
PR [ ]
MAKE "ｲﾁﾗﾝ [ ]
MAKE "ｼﾝｲﾁﾗﾝ [ ]
MAKE "ﾁｮｳﾌｸ [ ]
ﾆｭｳﾘｮｸ
ﾁｪｯｸ
ｼｭﾂﾘｮｸ
END

TO ｼｭﾂﾘｮｸ
CS
PR [♠♠ ｱﾀﾗｼｲ ﾘｽﾄﾃﾞｽ ♠♠]
PR :ｼﾝｲﾁﾗﾝ
PR [ ]
IF EMPTYP :ﾁｮｳﾌｸ [STOP]
PR [♦♦ ﾁｮｳﾌｸｼﾃｲﾀﾉﾊ ♦♦]
PR ( SE [==] :ﾁｮｳﾌｸ [ﾃﾞｼﾀ ==] )
END

TO ﾁｪｯｸ
MAKE "ｻﾝﾌﾟﾙ FIRST :ｲﾁﾗﾝ
IF MEMBERP "9999 :ｻﾝﾌﾟﾙ [STOP]
MAKE "ｲﾁﾗﾝ BUTFIRST :ｲﾁﾗﾝ
TEST MEMBERP :ｻﾝﾌﾟﾙ :ｲﾁﾗﾝ
IFFALSE [MAKE "ｼﾝｲﾁﾗﾝ LPUT :ｻﾝﾌﾟﾙ :ｼﾝｲﾁﾗﾝ]
IFTRUE [MAKE "ﾁｮｳﾌｸ LPUT :ｻﾝﾌﾟﾙ :ﾁｮｳﾌｸ]
ﾁｪｯｸ
END

TO ﾆｭｳﾘｮｸ
PR [ｵﾜﾘﾅﾗ 9999 ｦ ﾆｭｳﾘｮｸ]
MAKE "ﾋﾄﾘﾌﾞﾝ RL
MAKE "ｲﾁﾗﾝ LPUT :ﾋﾄﾘﾌﾞﾝ :ｲﾁﾗﾝ
IF MEMBERP "9999 :ﾋﾄﾘﾌﾞﾝ [STOP]
ﾆｭｳﾘｮｸ
END

MAKE "ﾁｮｳﾌｸ [[A] [B] [D] [B]]
MAKE "ｼﾝｲﾁﾗﾝ [[F] [G] [D] [A] [B]]
MAKE "ｲﾁﾗﾝ [[9999]]
MAKE "ｻﾝﾌﾟﾙ [9999]
MAKE "ﾋﾄﾘﾌﾞﾝ [9999]
```

▲リストから重複したものを取りのぞき，新しいリストを作る手続き

ファイルの作成はこうする

＊Logoとファイル

Logoを使って，データをフロッピーに収めることを考えましょう．Logoではプリンタやキーボード，ディスプレイなどもファイルを扱うデバイス（機器）として扱います．BASICではPRINT＃，INPUT＃あるいはPUT，GETといった命令を使って，バッファのデータをフロッピーに送ったり，フロッピーからデータをもらったりしていましたが，Logoではフロッピーに書いたり，フロッピーから読んだりする専用のプリミティブはありません．ディスクをオープンして，書きこみの準備（SETWRITE）や読出しの準備（SETREAD）をしたら，PRINTやREADLISTなど，いままではキーボードやディスプレイに対して命令していた入／出力のプリミティブを使います．

＊＊フロッピーへのセット方法は

サンプルは主手続きが“成績ファイルの作成”です．ディスクドライブの0番を使うように設定したら，“入力”の手続きを呼びます．“入力”の手続きでは名前と成績の2つの変数にそれぞれのデータを入力したら，“書込み”の手続きを呼んでいます．“書込み”では，セットライト文を書き，2つの変数をフロッピーに書込んでいきます．それが終ったら，もうひとつのセットライト文があります．ここで指定したデバイスは「S」……つまりスクリーンです．

“入力”の手続きの最後に“入力”を呼んでいるので，1回分の入力が終ったら，また入力を続けます．通常の入力を続けている限り，氏名と成績の2つのデータがどんどんフロッピーへと書込まれてゆきます．終らせたい時は「ZZZ」を入力します．

なお，作成したデータファイルは，BASICとは違ってディレクトリをとって見ても，プログラムなのかファイルなのかの区別はできません．ですからLogoで作ったデータ用のファイル名は，たとえばサンプルのようにファイル名のあとに「DT」などをつけたりして，識別できるようにしましょう．

データの記録方法はこのほかに，たとえば名前の変数をひとつのリスト形式にして（例：[山田　上田　吉岡……]），いちばん最後にまとめてフロッピーに書込むことも考えられます．なお，セットリード文は手続きの中でしか使えないので注意しましょう．

＊＊＊データを読む

データを読むための，いちばん簡単な方法は，D0を使うとして

POFILE "D0：ファイル名

になります．このプリミティブを使うときはOPEN文は不必要です．

ファイルされたデータは，やはりOPEN文を使って指定のファイルを開き，SETREADによって可能になります．多くのデータが入っているファイルを読んでゆくときはBASICとおなじように，データが最後かどうかを調べないといけません．ファイルの末尾を知るためにはEOFPを使います．データがある間は偽を，末尾なら真を返してきます．このEOFPとIFやTESTその他のプレディケイトとを組合せることで，指定のファイルに存在しているデータを取出したりすることができます．

ファイルを操作して目的のデータを探したり，ソートして新たなデータを作り，それをもういちどファイルにするといった作業も可能ですが，じっくりと考えないとかなり難しいかもしれません．

```
TO ｾｲｾｷﾌｧｲﾙｻｸｾｲ
TS 80CHAR
OPEN "D0:ｺｸｺﾃｽﾄDT
ﾆｭｳﾘｮｸ
END

TO ﾆｭｳﾘｮｸ
PR [ﾅﾏｴ ｦ ﾄﾞｳｿﾞ * ｵﾜﾘﾅﾗ ZZZ ｦ ﾆｭｳﾘｮｸ ｼﾃｸﾀﾞｻｲ]
MAKE "ﾅﾏｴ READLIST
IF MEMBERP "ZZZ :ﾅﾏｴ [ｵﾜﾘ STOP]
PR [ｾｲｾｷ ｦ ﾄﾞｳｿﾞ]
MAKE "ｾｲｾｷ READLIST
PR [ ]
ｶｷｺﾐ
ﾆｭｳﾘｮｸ
END

TO ｵﾜﾘ
PR [ﾆｭｳﾘｮｸ ﾊ ｵﾜﾘﾃﾞｽ * * * ﾄｯﾌﾟﾚﾍﾞﾙﾆ ﾓﾄﾞﾘﾏｽ * * *]
CLOSE "D0:ｺｸｺﾃｽﾄDT
END

TO ｶｷｺﾐ
SETWRITE "D0:ｺｸｺﾃｽﾄDT
PRINT :ﾅﾏｴ
PRINT :ｾｲｾｷ
SETWRITE "S:
END
```

Logoによるソートの考えかた，作りかた

＊Logoにおける“並べかえ”について

コンピュータがもっとも得意にしているものに“ならべかえ”……ソーティングがあります．Logoにも，もちろんできますが，BASICとは考えかたを変えなければいけません．というよりは，考えかたはおなじでも，書法が異るといったほうがよいでしょう．Logoによる数値のソーティングを考えてみましょう．なお，サンプルのプログラムはなるべく分りやすいように，手続きを細かくしています．

＊＊3つの数を例にして考えてみよう

まず具体的なリストを例にして考えましょう．いま，つぎのリストがあるとします．

[3 2 5]

実際の入力作業では，入力のさいごに“Z”を入れてそれを合図にして，つぎの手続きに入りますし，このZを利用するためにリストのさいごにこれを入れておきます（“入力”の手続き参照）．

：リスト＝[3 2 5 Z]

“入力”の次の手続きは“取出し”です．ここでは

MAKE ″MIN FIRST ：リスト

（リストの最初の要素を取出して，：MINに代入しなさい）

MAKE ″リスト BUTFIRST ：リスト

（リストの最初以外の要素を取出して，：リストに代入しなさい）

ですから，：MIN＝3　：リスト＝[2　5　Z]　になります．

次は“並べかえ”の手続きです．

IF EMPTYP ：リスト [シュツリョク]

（：リストが空なら，“出力”の手続きを呼びなさい）

：リストは空ではないので，次へゆきます．

MAKE ″サンプル FIRST ：リスト

（：サンプルに，：リストの最初の要素を取出して代入しなさい）

ですから，：サンプル＝2になります．

IF ：サンプル＝″Z [操作 取出し 並べかえ]

（取り出した：サンプルがZなら，3つの手続きを呼びなさい）

：サンプルはZではないので，ここを無視します．つぎは

MAKE ″リスト BUTFIRST ：リスト

なので，：リスト＝[5Z] になります．

IF ：MIN ＞ ：サンプル ［入れかえ］

（もし：MIN（実際には3）が：サンプル（実際には2）よりも大きか

```
TO ｿｰﾄ
ｼｮｷｾｯﾃｲ
ﾆｭｳﾘｮｸ
ﾄﾘﾀﾞｼ
ﾅﾗﾍﾞｶｴ
ｼｭﾂﾘｮｸ
END

TO ﾅﾗﾍﾞｶｴ
IF EMPTYP :ﾘｽﾄ [ｼｭﾂﾘｮｸ]
MAKE "ｻﾝﾌﾟﾙ FIRST :ﾘｽﾄ
IF :ｻﾝﾌﾟﾙ = "Z [ｿｳｻ ﾄﾘﾀﾞｼ ﾅﾗﾍﾞｶｴ]
MAKE "ﾘｽﾄ BUTFIRST :ﾘｽﾄ
IF :MIN > :ｻﾝﾌﾟﾙ [ｲﾚｶｴ]
MAKE "ﾘｽﾄ LPUT :ｻﾝﾌﾟﾙ :ﾘｽﾄ
ﾅﾗﾍﾞｶｴ
END

TO ｲﾚｶｴ
MAKE "ﾀﾞﾐｰ :ｻﾝﾌﾟﾙ
MAKE "ｻﾝﾌﾟﾙ :MIN
MAKE "MIN :ﾀﾞﾐｰ
END

TO ﾄﾘﾀﾞｼ
MAKE "MIN FIRST :ﾘｽﾄ
MAKE "ﾘｽﾄ BUTFIRST :ﾘｽﾄ
END

TO ｿｳｻ
MAKE "ｿｰﾄﾘｽﾄ LPUT :MIN :ｿｰﾄﾘｽﾄ
MAKE "ﾘｽﾄ LPUT FIRST :ﾘｽﾄ :ﾘｽﾄ
MAKE "ﾘｽﾄ BUTFIRST :ﾘｽﾄ
END

TO ｼｭﾂﾘｮｸ
PR [ ]
PR ( SE [◆◆ ﾓﾄﾉ ﾘｽﾄ |ﾃﾞｰﾀｽｳ =] COUNT :ﾓﾄﾉﾘｽﾄ [| ◆◆] )
PR :ﾓﾄﾉﾘｽﾄ
PR [ ]
PR [◆◆ ﾅﾗﾍﾞｶｴ ｼｭｳﾘｮｳ ﾘｽﾄ ◆◆]
PR :ｿｰﾄﾘｽﾄ
ﾂｷﾞﾉｿｰﾄ
END

TO ﾂｷﾞﾉｿｰﾄ
PR [ ]
PR [ﾏﾀ ｿｰﾄ ｦ ｻｾﾏｽｶ ?]
PR [ｽﾙ････ > Y ｼﾅｲ BREAK KEY]
IF "Y = FIRST RL [ｿｰﾄ]
ﾂｷﾞﾉｿｰﾄ
END

TO ﾆｭｳﾘｮｸ
PR [ｶｽﾞｦﾄﾞｳｿﾞ ｵﾜﾘﾅﾗ Z ｦ ﾆｭｳﾘｮｸ]
MAKE "X FIRST RL
MAKE "ﾘｽﾄ LPUT :X :ﾘｽﾄ
IF "Z = :X [STOP]
MAKE "ﾓﾄﾉﾘｽﾄ :ﾘｽﾄ
ﾆｭｳﾘｮｸ
END

TO ｼｮｷｾｯﾃｲ
80CHAR
TS
CS
MAKE "ﾘｽﾄ [ ]
MAKE "ｿｰﾄﾘｽﾄ [ ]
END

MAKE "ﾘｽﾄ [ ]
MAKE "ｻﾝﾌﾟﾙ "Z
MAKE "MIN "Z
MAKE "ｿｰﾄﾘｽﾄ [2 3 5]
MAKE "ﾓﾄﾉﾘｽﾄ [3 2 5]
MAKE "X "Z
MAKE "ﾀﾞﾐｰ 2
```

▲ソーティングの手続き（主手続きは“ソート”）

ったら，“入れかえ”の手続きを呼びなさい）

3＞2は真（TRUE）ですから，“入れかえ”をします．入れかえたあとは：MIN＝2　：サンプル＝3になります．つぎは

MAKE ″リスト LPUT ：サンプル ：リスト

（リストという変数を，：リストのさいごに：サンプルをつけたものにしなさい）

：サンプルは3，リストは［5　Z］でしたから，新しいリストは

：リスト＝[5 Z 3]

になります．最初，いちばん最後にあったZがひとつ，前へ進んでいることに注意してください．

＊＊＊“Z”の位置で並べかえを判定させる

こうして，また並べかえにもどります．

：MIN＝2 ：サンプル＝5 ：リスト＝[5 Z 3]

の状態で，：MINと：サンプルを比べますが，：MINのほうが小さいので，ここでは入れかえをせずに，サンプルをリストのさいごにつけます．したがって2回目の操作では

：MIN＝2 ：リスト＝[Z 3 5]

になります．こうしてまた“並べかえ”になりますが，3回目の時に：サンプル＝Z

になるので，まず“操作”を呼びます．

MAKE ″ソートリスト LPUT ：MIN ：ソートリスト

（結果的には：ソートリスト＝［2］になります）

MAKE ″リスト LPUT FIRST ：リスト ：リスト

（リストという変数を，：リストの最初の要素を取出して，それを：リストという変数のさいごにつけなさい）

この手続きに入る前の：リスト＝［Z　3　5］でしたから，この命令によって新しいリスト＝［Z　3　5　Z］になります．このあと“操作”の最後の命令を実行します．

MAKE ″リスト BUTFIRST ：リスト

（これによってリスト＝［3　5　Z］になります）

これでリストの中にある，いちばん小さい数が“ソートリスト”に入り，しかもいちばん始めはリストに入っていた数が消えます．つぎに“取出し”の手続きですから：MIN＝3になり，リスト＝［5　Z］になって，また“並べかえ”にゆくというわけです．

用語総索引

50音順，ＡＢＣ順

注）コマンドやプリミティブ，慣用的に大文字で表わされるものを大文字，一般用語などは小文字にしている．
太字はLogoに関係するもの．

掲載主要プログラム

BASIC

Logo

著 者 略 歴

上柿　力（うえがき つとむ）

1943年　千葉県銚子生まれ

1967年　東京電機大学電子工学科卒業

1967年　株式会社ラジオ技術社入社

「ステレオ芸術」,「パソコン実用ガイド」編集長などを経て，現在著述業．

著　書　「FM-7グラフィックBOOK」大河出版

「親子で楽しむMSX」ラジオ技術社

FM-77の本——BASIC, Logoプログラミング——　定価 2,200円

昭和59年 9月20日　初 版 発 行

検 印
省 略

Printed in Japan

著　者　上 柿　　力
編集人　鈴 木 勇 治
発行者　金 井　　稔

発行所 株式会社 ラジオ技術社
〒101　東京都千代田区神田淡路町2－1
☎03(251)0498, 1321〈振替・東京5-2506〉

落丁本，乱丁本はお取替いたします．

三美印刷　トキワ製本

ラジオ技術選書148

FM-77の本

1 これだけは知っておきたいFM-77のABC
2 本格的なプログラム作りのために
3 すっかり分るファイルの操作
4 グラフィックスをマスターしましょう
5 グラフィックスの進んだ使いかた
6 漢字の取扱いとその応用
7 サウンド&ミュージック
8 これからの言語Logoを研究しよう

ラジオ技術社
定価 2,200円
ISBN4-8443-0148-9 C2055 ¥2200E